Cost-Driven Build Orientation and Bin Packing of Parts in Selective Laser Melting (SLM)

Valeriya Griffiths, James P. Scanlan, Murat H. Eres[1]

University Technology Centre for Computational Engineering, University of Southampton, Southampton, SO16 7QF, UK

Antonio Martinez-Sykora

Southampton Business School, University of Southampton, Southampton, SO17 8BJ, UK

Phani Chinchapatnam

Rolls-Royce plc, PO Box 31, Derby, DE24 8BJ, UK

Abstract

Selective Laser Melting (SLM) is an additive manufacturing process capable of producing mixed batches of parts simultaneously within a single build. The build orientation of a part in SLM is a key process parameter, affecting the build cost, time and quality, as well as batch size. Choosing an optimal arrangement of multiple heterogeneous parts inside the SLM machine also presents a challenging irregular bin packing problem. Since the two problems are interdependent, this paper addresses the combined problem of finding an optimal build orientation and two-dimensional irregular bin packing solution of a mixed batch of parts across identical SLM machines. We address this problem specifically in the context of low-volume high-variety (LVHV) production in the aerospace sector, using total build cost as the objective function. To solve this problem, we present an Iterative Tabu Search Procedure (ITSP), which consists of six distinct stages. We test each stage in the ITSP on 27 manually generated instances, based on 68 unique geometries ranging in convexity and size, including six real-life components from the aerospace industry. Two of the six stages, which are driven by support structure volume, returned the highest improvement in cost. Overall, the results showed an average cost improvement of 16.2% over the initial solution. The initial solution of the procedure was benchmarked against a commercial software, showing comparable results.

Keywords: Selective Laser Melting (SLM), Additive Layer Manufacturing (ALM), 2D Irregular Bin Packing Problem (2DIBPP), Tabu Search.

1. Introduction

This paper addresses the problem of arranging complex parts inside Selective Laser Melting (SLM) machines. SLM belongs to a family of methods known as additive layer manufacturing (ALM), which create parts additively (as opposed to subtractively) in a layer-upon-layer manner, directly from a Computer-Aided Design (CAD) file. While originally, ALM methods were used solely for Rapid Prototyping (RP), the quality of these methods has improved significantly over the past two decades, making them suitable for the production of high quality functional parts.

[1] vg2g10@soton.ac.uk

Our interest in SLM comes from its ability to produce highly complex and lightweight parts from common industrial materials such as steel, titanium, nickel and aluminium alloys, making it an attractive technology for aerospace, automotive and biomedical applications (Frazier, 2014; Guo & Leu, 2013). The SLM process is shown schematically in Figure 1 and works in the following steps:

1. A thin layer of powder, typically 20-50 microns, is spread over the build platform by a recoater blade;
2. A moving laser selectively scans the build platform, locally fusing the powder particles to create a layer of the part;
3. The build platform steps down and a new layer of fresh powder is distributed evenly across the platform;
4. Steps 1-3 are repeated until the part is fully formed.

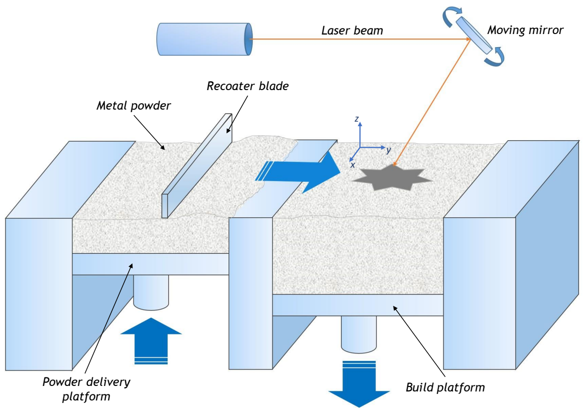


Figure 1: The Selective Laser Melting (SLM) process. The build direction is defined by the z-axis.

Unlike most other manufacturing techniques, SLM creates parts directly from the powder material without the need for tooling. Therefore, since SLM is not bound by traditional manufacturability constrains (for instance, the need for tool access), it can achieve very complex geometries that would be infeasible or too expensive to produce otherwise. Moreover, SLM offers a more competitive unit cost for parts produced in small volumes, and can produce a wide range of geometries in a single build (i.e. a mixed batch) at no extra cost; in contrast, traditional manufacturing tools are often tailored to a specific product or family of parts and production volumes must be sufficiently high to justify the cost of these tools (Hopkinson & Dickens, 2003; Atzeni & Salmi, 2012).

1.1. Part Build Orientation

The three-dimensional (3D) rotational position of a part inside the SLM machine is known as the build orientation, and has a significant impact on the build cost, build time, scrap material and surface quality of that part. Since a part can be produced in many different build orientations, choosing the best one presents an important multi-objective decision problem in SLM process planning.

A common feature shared by several ALM methods, is the need for sacrificial support structure. In SLM, the purpose of this support structure is two-fold: firstly, the part must be stabilised and supported against the forces of gravity; secondly, the additional solid material helps to conduct away heat and reduce the residual thermal stresses within the part, while counteracting the distortive effects of these stresses. As a result, it is currently very difficult to completely avoid the need for support structure in SLM (Jhabvala et al., 2011). However, it is desirable to minimise the volume of necessary support structure as it incurs additional material costs, prolongs the build time and requires post-build removal. It is common practice to place support structures underneath all overhanging downward-facing surfaces exceeding a 45° angle with respect to the vertical axis (Thomas, 2009; Järvinen et al., 2014; Calignano, 2014). Thus, a common objective used to search for candidate build orientations is to minimise the total area of such surfaces (Alexander et al., 1998; Canellidis et al., 2009). An alternative objective is to minimise the projected volume of support structures as demonstrated Das et al. (2015) and Schwerdt et al. (2000), which provides a more accurate reflection of the scrap material cost, although the additional intersection checks incur a significant computational cost.

Other criteria for evaluating build orientations include surface roughness (Ahn et al., 2007), build time and build cost (Lan et al., 1997; Byun & Lee, 2006). The main challenge comes from the multi-objective nature of this problem, which not only increases the size and complexity of the solution space, but also makes it difficult to define a suitable objective function to encapsulate all of the above criteria. This is reflected by the lack of agreement in existing literature on this topic, as different papers propose different objective functions. A number of papers have formulated the objective function as a weighted sum of the criteria (Byun & Lee, 2006; Canellidis et al., 2009), while several others chose to optimise a single primary criterion, using other criteria to break tie (Frank & Fadel, 1995; Alexander et al., 1998); a third approach is to provide a Pareto front and leave it to the user to select a preferred non-dominated solution (Padhye & Deb, 2011).

Furthermore, when placing a batch of mixed parts inside a SLM machine, the build orientation of each part determines the number of parts that can fit inside the machine, as well as the build cost and build time of the whole batch (Rickenbacher et al., 2013). Only Zhang et al. (2015a) consider the build orientation of multiple parts at once, as well as grouping parts into batches based on their similarities, but do not consider bin packing. Their method uses a genetic algorithm (GA) to search for the closest solution matching a set of aspirational target values provided by the user, for instance, a desired maximum height and projected area of parts.

1.2. Part Arrangement in ALM

Despite the absence of tooling costs, cost amortisation can still be achieved in SLM through efficient machine utilisation, as demonstrated in the literature (Ruffo & Hague, 2007; Atzeni & Salmi, 2012; Piili et al., 2015). The need for an optimal part arrangement to maximise machine utilisation, presents what is known as a bin packing problem. A number of methods have been proposed to solve this problem, both in 2D (Canellidis et al., 2006, 2013; Zhang et al., 2016a) and in 3D (Ikonen et al., 1997; Wodziak & Fadel, 1999; Gogate & Pande, 2008; Wu et al., 2014). In all of these methods the parts are packed into a single bin, either by pre-selecting a sufficiently small group of parts (Zhang et al., 2016a), or by solving a Knapsack problem (Canellidis et al., 2013), with the objective of packing as many parts as possible inside the available space. The choice between 2D and 3D bin packing is mainly determined by the ALM method. In SLM, the presence of high residual stresses and the need for support structure means that stacking parts vertically is likely not only to increase the build cost and complexity of support structure, but also lead to an increased risk of build failure and damage during the removal of parts from the build plate (Zhang et al., 2016a). This also applies to other ALM methods such as Stereolithography. Therefore, SLM and Stereolithography machines are typically treated as 2D bins. Conversely, 3D packing is acceptable in methods such as Selective Laser Sintering (SLS), since the residual stresses are much lower and parts can often be supported by the surrounding powder without the need for additional solid supports (Leary et al., 2014).

Most of the literature related to this problem agrees on the important role played by part build orientation, and a number of authors (Wodziak & Fadel, 1999; Canellidis et al., 2006; Wu et al., 2014; Zhang et al., 2016a) have tried to optimise the build orientation, albeit in a separate step prior to bin packing. Conversely, Ikonen et al. (1997) allow 3D rotation of parts to improve their bin packing solution for SLS, however, each part is limited to 24 predefined build orientations, and typical build orientation objectives (e.g. build cost) are not considered.

Only Zhang et al. (2017) have addressed the combined problem of build orientation and bin packing of heterogeneous parts. However, as with previous works Zhang et al. (2017) consider the problem with a single bin and use part overlap and packing density as their driving optimisation criteria.

1.3. Research Motivation

The geometric flexibility of SLM and its ability to produce parts in mixed batches, make it particularly advantageous for high-variety low-volume (HVLV) production, for example, customised components in the medical sector and spare parts in the aerospace sector Thomas (2016). According to Ruffo & Hague (2007) the production of mixed batches is already common practice employed by RP bureaus and specialised ALM service providers, who often produce small quantities of parts of various sizes, functions and complexities for a number of different customers. Demands of aircraft spare parts are particularly difficult to forecast (Ghobbar, 2004); in practice, this often leads to overstocking, high inventory costs and risk of delays. Thus, a number of researchers (Khajavi et al., 2014; Holmström et al., 2010) suggest on-demand production of spare parts via SLM as a better alternative.

We therefore consider the following scenario. The short-term (i.e. on the order of days and weeks, rather than months and years) demands of different spare parts are lumped together based on their required delivery times, material and assemblies, and produced in mixed batches via SLM. These demands are assumed to be fixed once the internal orders are placed and overproduction of parts is not allowed. This presents a challenging problem, which combines the build orientation of multiple parts and bin packing of heterogeneous and irregular shapes.

We also consider the presence of identical parts in the same batch. If two identical parts are built in two different build orientations, each part can be expected to exhibit different surface qualities, dimensional accuracy and mechanical properties (Ahn et al., 2007; Chlebus et al., 2011). Aircraft parts must go through a time-consuming and expensive certification process and meet stringent airworthiness requirements (Saha et al., 2013). If a change in the manufacturing route leads to different part properties, that part must be re-certified. Thus, even if the additional manufacturing variability is deemed acceptable by the user, the cost of certifying a part for each different build orientation is likely to outweigh any benefits. For this reason, we constrain the problem so that identical geometries can only be placed in the same build orientations.

Additionally, the following gaps have been identified in existing literature related to this problem:

- Because the build orientation is a multi-objective problem, there is currently a lack of agreement on what is the most suitable objective function. Existing multi-objective functions often require the user to provide appropriate weight coefficients or target values, which is a difficult task in practice.

- Existing bin packing approaches are limited to a single bin, and most restrict the movement and rotation of parts. Additionally, no works have been found to consider the presence of build plate fixtures, which affect the bin packing problem in a non-trivial way.

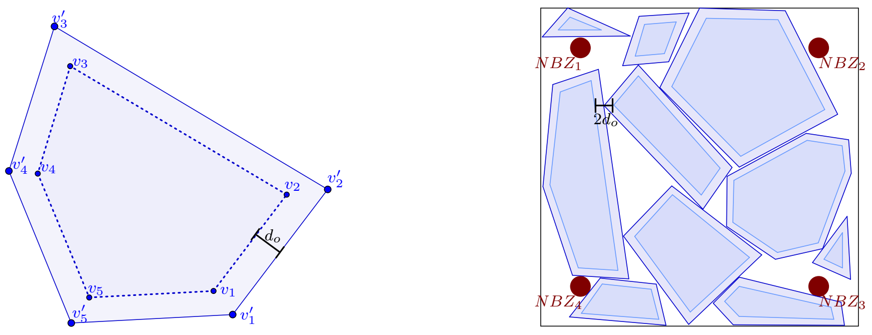
- Most importantly, no works have addressed the combined problem of build orientation and bin packing, with the exception of Zhang et al. (2017). However, even this work only addresses the bin packing problem in a single bin (i.e. without considering bin assignment) and does not consider the effects of build orientations on the cost of parts – two realistic and very important considerations.

To address these gaps, we propose build cost as the objective function, while using secondary parameters, such as build height, support structure volume and packing efficiency to drive our heuristic. Our view is that any parameter affected by build orientation can in fact be translated into a cost element; for instance, support structure volume can be treated as a scrap material cost; build height is a direct driver of build time and build cost; while insufficient quality and dimensional accuracy can be considered as an additional post-processing or scrap cost. Furthermore, product cost is the core driver of any business and industries are unlikely to adopt SLM over traditional manufacturing techniques unless it is cost-competitive. Thus, a cost-driven process planning tool can be used not only to optimise the SLM process parameters but to quickly compare it against other manufacturing routes.

2. Problem Description

Let $G = \{1, \ldots, N\}$ be a set of N unique 3D geometries, which are to be placed into identical SLM machines. The width, length and height of the build envelope within each machine are denoted by W, L and H, respectively, and it is assumed that enough machines are available to accommodate all provided geometries. It is also assumed that each geometry $g \in G$ has a demand q_g, to account for non-unique geometries. In this paper, vertical stacking of parts in the SLM machines is forbidden (for reasons discussed in Section 1.2), therefore, the problem can be decomposed into the following two sub-problems.

Firstly, a favourable build orientation must be selected for each geometry $g \in G$, following which, a 2D polygon of g is projected onto the horizontal plane. Throughout the rest of this paper we refer to this 2D projection as a 'piece', denoted by p_g. The quality of each build orientation is determined by the geometry height, total support structure volume and area of the resulting piece. The geometry height may not exceed H, while the area of p_g may not exceed $W * L$. The output of this stage is a set of unique pieces $P = \{p_1, \ldots, p_N\}$, where each piece corresponds to a geometry g and demand q_g. This means that g can only have one build orientation at a time, satisfying the 'certification condition' outlined in Section 1.3.



(a) Example of a projected polygon (dotted line), inflated by a constant offset d_o to create the final piece (solid line).

(b) Example of packed inflated pieces shown in blue and no-build zones (NBZ) shown in red.

Figure 2: Additional SLM constraints in the 2D bin packing problem

Secondly, a 2D irregular bin packing problem (2DIBPP) must be solved for a total of $\sum_1^N q_g$ pieces, by assigning and packing all pieces in P in their corresponding demands into a minimum number of identical bins of width W and length L. To avoid part damage during the SLM process and to facilitate post-build removal of parts, a small gap must be maintained between the parts when they are placed on the build plate. This condition can be met by inflating the pieces by a fixed offset, d_o, as shown in Figure 2a. As indicated in Figure 2b, inflating all pieces by d_o produces a minimum gap of $2d_o$ between parts. Figure 2b also shows four no-build zones, which represent four bolts used to fix the build plate, as is done in most typical SLM machines (although the exact configuration depends on the machine manufacturer).

A solution to the problem would be given by a set of bins $B = \{b_1, \ldots, b_M\}$, where M is the total number of bins. Each bin is denoted by $b_i(P_i, O_i, X_i, Y_i)$, where $P_i \subseteq P$. During the search for a packing solution the pieces are allowed to rotate continuously about the vertical axis, hence, $O_i = \{o_1, \ldots, o_n\}$ defines the 2D orientation of each piece. Finally, $X_i = \{x_1, \ldots, x_n\}$ and $Y_i = \{y_1, \ldots, y_n\}$ define the x and y coordinates of the reference point of each piece, respectively. The reference point of a piece is the bottom left corner of its enclosing rectangle, corresponding to its 2D orientation. Every piece in P must be placed completely inside the bins and may not overlap with any other pieces or the no-build zones, as shown in Figure 2b.

The objective is to find a solution with M bins which yields the minimum overall build cost, as shown in Equation 1. A number of cost models have been proposed for SLM (Ruffo & Hague, 2007; Baumers et al., 2016; Hopkinson & Dickens, 2003; Rickenbacher et al., 2013). In this paper we use the model proposed by Ruffo & Hague (2007) due to its simplicity and its focus on low to medium production volumes. This model uses a popular model structure that splits the build cost into two main components: indirect cost, which includes administrative, software and shop floor space costs; and direct cost, which includes resources directly consumed by the process, in this case, the material. This model is adapted in Equation 2, showing the build cost c_i for bin b_i.

$$\text{min.} \sum_{i=1}^{M} c_i \tag{1}$$

$$c_i = C_1 t_i + C_2 \rho \sum_{k=1}^{n_i} (v_k + s_k) \tag{2}$$

where C_1 and C_2 are coefficients corresponding to indirect cost (£/h) and material cost (£/kg), respectively; t_i is the build time (h) of bin b_i; and ρ is the material density ($\mathrm{kg\,mm^{-1}}$). The number of pieces in b_i is denoted by n_i, where each piece k has a geometry volume v_k ($\mathrm{mm^3}$), and support structure volume s_k ($\mathrm{mm^3}$).

The build time of SLM can be modelled as a function of many different parameters, such as part dimensions, layer thickness, hatching and laser settings (Rickenbacher et al., 2013; Zhang et al., 2015b). In this paper we use a simple linear regression model, shown in Equation 3, to approximate the build time as a function of total part volume, total support structure volume and build height, which is the height of the tallest geometry in the bin.

$$t_i = r_1 + r_2 h_i + r_3 \sum_{k=1}^{n_i} v_k + r_4 \sum_{k=1}^{n_i} s_k \tag{3}$$

where r_1 is the regression model constant (h); h_i is the build height of bin b_i; r_2 is the build height coefficient ($\mathrm{h\,mm^{-1}}$); and r_3 and r_4 are the total part volume and support structure volume coefficients ($\mathrm{h\,mm^{-3}}$), respectively. The details of this model can be found in Appendix A.

3. Iterative Tabu Search Procedure (ITSP)

From Equations 1-3, we can observe that the total build cost for a batch of parts is driven by the number of bins, the build height, and the total support structure volume of each bin (since part volume is constant). As mentioned above, the problem must consider not only the build orientation of each unique geometry $g \in G$, but also the bin assignment and bin packing solution of their corresponding pieces. This results in a very large and complex solution space, and what is known as an NP-hard problem. Currently, it is infeasible to solve NP-hard problems in reasonable time by any exhaustive approaches, and so a carefully designed heuristic approach is required.

For this purpose, we develop a new heuristic, which aims to guide the optimisation search to different promising neighbourhoods of the solution space, by focusing on different cost factors in the objective function described in Equations 1-3 – namely, the number of bins, the height of each bin and the support structure volume. To properly reduce each cost factor, we must address it directly in a separate stage because it is affected by several pieces in the solution. For example, to minimise the height of one of the bins in the solution, it is not enough to reduce the height of only one piece in that bin; instead we try to reduce the height of the tallest piece in the bin repeatedly until we cannot reduce the bin height anymore without increasing the number of bins. Similarly, reducing the area of only one piece is not likely to reduce the number of bins in the solution; thus, we reduce the area of all pieces when attempting to find the minimum number of bins. Additionally, the bin assignment of pieces also has a significant effect on the number of bins and their height, thus, we have additional stages where bin assignment is modified to see if the cost can be reduced further.

Our approach is called the Iterative Tabu Search Procedure (ITSP) and consists of six distinct stages. The key steps in the ITSP are summarised in Figure 3, where G denotes a set of geometries, B denotes a set of bins in the solution and P denotes a set of pieces, where each piece p has a corresponding area, height and support structure volume denoted by a_p, h_p and s_p, respectively. Pieces which have been improved by the Tabu search are denoted with an asterisk (e.g. p^*).

In the first stage of the ITSP we generate the initial solution and try to fit all geometries into a minimum number of bins as mentioned above. This is achieved in the following steps. Firstly, an initial set of 2D pieces P is generated for the input set of geometries G. A Tabu search (outlined in Section 5) is performed on each piece $p \in P$, to find an improved piece p^* with the minimum piece area a^*. Finally, a 2DIBPP is solved for the resulting set P^*, producing a solution with the smallest set of bins B. The remaining five stages attempt to improve the cost of each bin in B, as follows:

- Stage 2, to which we refer to as Single Bin Build Height Reduction (SHR), reduces the build height of partially full bins, without disrupting the piece assignment.
- Stage 3, Single Bin Support Structure Volume Reduction (SSVR), reduces the support structure volume of each pieces inside each partially full bin, without disrupting the piece assignment.
- Stage 4, Pairwise Bin Build Height Reduction (PHR), reduces the build height of bins in pairs, allowing pieces to move between the bins.
- Stage 5, Pairwise Bin Support Structure Volume Reduction (PSVR), reduces the support structure volume of bins in pairs, allowing pieces to move between the bins.

- Stage 6, Bin Addition and Random Reassignment (BARR), opens a new empty bin and moves pieces selected randomly from the current solution into the new bin. The remaining pieces in the disrupted bins are re-oriented and repacked to reduce the cost of those bins.

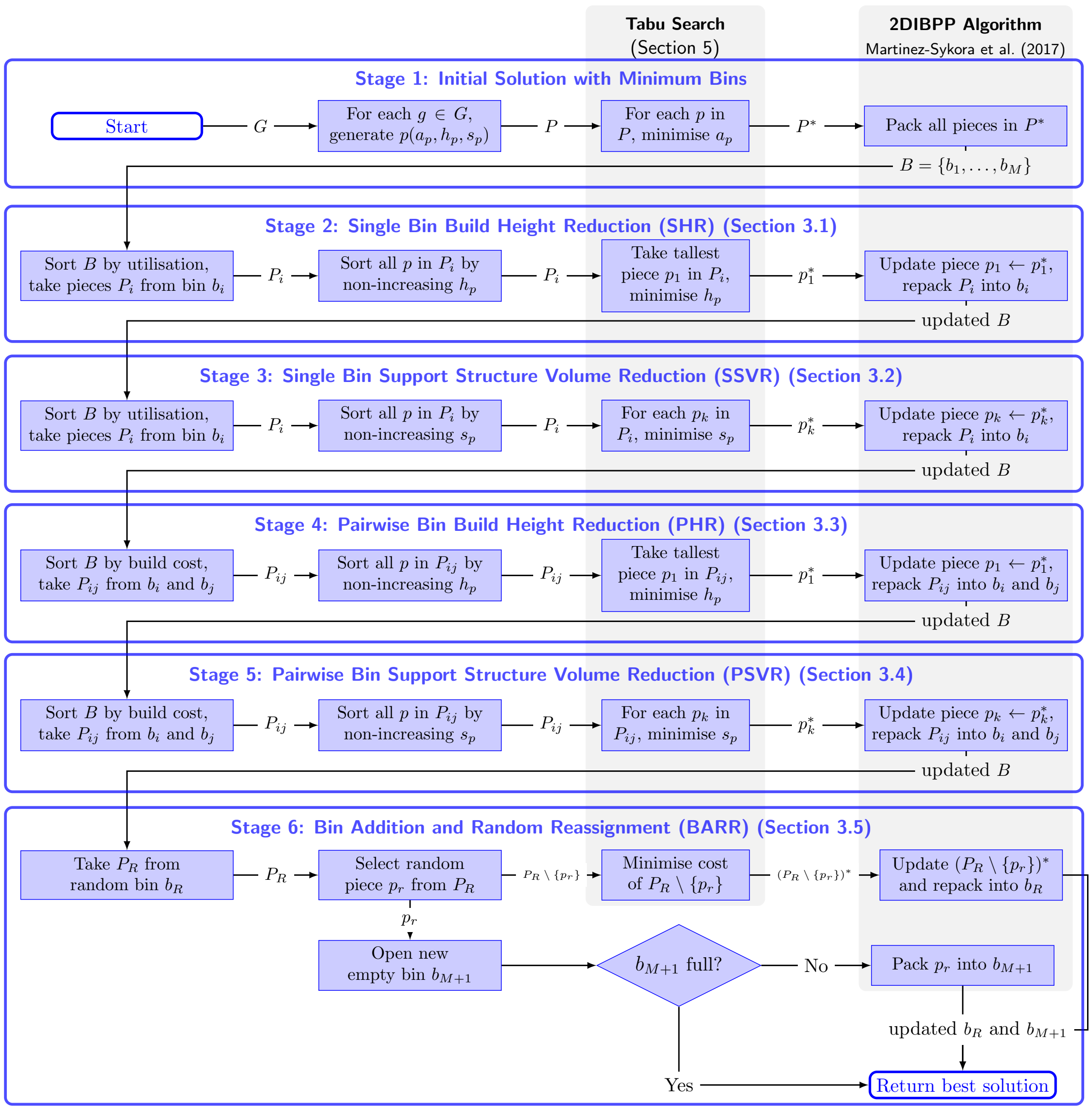

Tabu Search
(Section 5)
2DIBPP Algorithm
Martinez-Sykora et al. (2017)
Stage 1: Initial Solution with Minimum Bins
Start
G
For each $g \in G$, generate $p(a_p, h_p, s_p)$
P
For each p in P, minimise a_p
P^*
Pack all pieces in P^*
$B = \{b_1, \ldots, b_M\}$
Stage 2: Single Bin Build Height Reduction (SHR) (Section 3.1)
Sort B by utilisation, take pieces P_i from bin b_i
P_i
Sort all p in P_i by non-increasing h_p
P_i
Take tallest piece p_1 in P_i, minimise h_p
p_1^*
Update piece $p_1 \leftarrow p_1^*$, repack P_i into b_i
updated B
Stage 3: Single Bin Support Structure Volume Reduction (SSVR) (Section 3.2)
Sort B by utilisation, take pieces P_i from bin b_i
P_i
Sort all p in P_i by non-increasing s_p
P_i
For each p_k in P_i, minimise s_p
p_k^*
Update piece $p_k \leftarrow p_k^*$, repack P_i into b_i
updated B
Stage 4: Pairwise Bin Build Height Reduction (PHR) (Section 3.3)
Sort B by build cost, take P_{ij} from b_i and b_j
P_{ij}
Sort all p in P_{ij} by non-increasing h_p
P_{ij}
Take tallest piece p_1 in P_{ij}, minimise h_p
p_1^*
Update piece $p_1 \leftarrow p_1^*$, repack P_{ij} into b_i and b_j
updated B
Stage 5: Pairwise Bin Support Structure Volume Reduction (PSVR) (Section 3.4)
Sort B by build cost, take P_{ij} from b_i and b_j
P_{ij}
Sort all p in P_{ij} by non-increasing s_p
P_{ij}
For each p_k in P_{ij}, minimise s_p
p_k^*
Update piece $p_k \leftarrow p_k^*$, repack P_{ij} into b_i and b_j
updated B
Stage 6: Bin Addition and Random Reassignment (BARR) (Section 3.5)
Take P_R from random bin b_R
P_R
Select random piece p_r from P_R
$P_R \setminus \{p_r\}$
Minimise cost of $P_R \setminus \{p_r\}$
$(P_R \setminus \{p_r\})^*$
Update $(P_R \setminus \{p_r\})^*$ and repack into b_R
p_r
Open new empty bin b_{M+1}
b_{M+1} full?
No
Pack p_r into b_{M+1}
updated b_R and b_{M+1}
Yes
Return best solution

Figure 3: Schematic overview of the six stages in the ITSP. In each stage, key steps are shown in blue boxes, while the steps which use Tabu search and 2D bin packing are highlighted by a grey background.

The above stages are outlined in detail in Sections 3.1,3.2,3.3,3.4 and 3.5, respectively. In each stage, we employ the Tabu search outlined in Section 5 to search for build orientations. The process of generating build orientations is described in Section 4. To solve the 2DIBPP, we use the algorithm proposed by Martinez-Sykora et al. (2017). The algorithm consists of the following key steps: firstly, pieces are assigned to bins based on a naive 1D bin packing solution; secondly, a Mixed Integer Programming (MIP) model is computed in order to place the 2D pieces inside each bin. Because of the naive bin assignment the packing often fails, so a third and final step in the algorithm is to mend the assignment of pieces to each bin, until a feasible solution is found. This process is repeated until all remaining pieces have been packed successfully.

3.1. Single Bin Build Height Reduction (SHR)

A typical initial solution can be expected to contain at least one bin with very low utilisation as shown in Figure 4a; throughout this paper we refer to such bins as 'weak'. Unlike in many traditional cutting and packing (C&P) problem applications, such as the ceramic tile and sheet metal industries, weak bins do not offer any benefit in the context of SLM, since the un-utilised machine space cannot be recycled after the build. Instead, the build orientation of pieces inside the weak bin can be changed to minimise their build height, as shown in Figure 4b. Since build height is a driver of build time and cost, reducing the height of pieces improves the quality of the solution, and makes better use of the bin space. The SHR stage exploits this scenario.

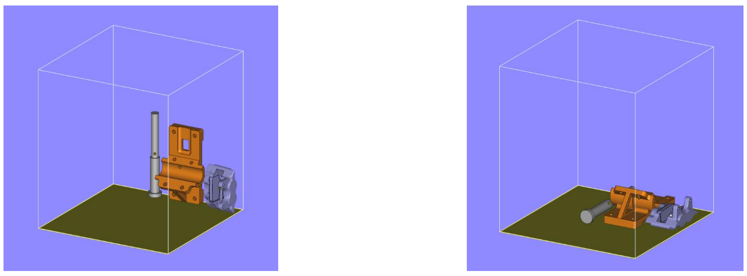

(a) Weak bin with very low utilisation. (b) Build orientation of each piece is changed to minimise build height.

Figure 4: Schematic example of a weak bin (left) that has been modified by the SHR stage (right).

As shown in Figure 3, the set of bins $B = \{b_1, \dots, b_M\}$ given by the initial solution is sorted by non-increasing utilisation, denoted by U, and only strong bins (e.g. with $U < 85\%$) are considered. For each bin b_i, the set of pieces $P_i = \{1, 2, \dots, n\}$ is sorted by non-increasing height. The tallest piece p_1 is selected and a Tabu search is performed to minimise the height of p_1. Let p_1^* denote the new piece returned by the Tabu search. If the height of p_1^* is lower than the height of p_1, we update $p_1 \leftarrow p_1^*$ in the set P and pass it to the 2DIBPP algorithm to be repacked into its original bin. The updated 2DIBPP solution is only accepted if the number of bins is not increased.

Depending on the demand q_g of the current piece, there may be several identical copies of piece p_1 in the solution. For this reason, P_i contains only unique pieces, and has a corresponding set of quantities $Q_i = \{q_1, q_2, \dots, q_n\}$, where q is the number of duplicate pieces p in b_i. Thus, any duplicates of p_1 inside bin b_i, can be updated easily by replacing p_1 with p_1^* in P_i. Furthermore, an additional check is performed on all bins in $B \setminus b_i$, where each bin that contains a duplicate of p_1 must also be updated and repacked.

If all bins containing p_1 are repacked successfully, the solution is updated and the process repeats, until no further height reduction can be achieved in b_i. If the SHR stage fails to achieve a height improvement, either during the Tabu search or during repacking, the stage will move on to the next bin in B. The stage terminates when all bins in B with $U < 85\%$ have been explored.

Finally, if a bin contains several pieces of the same height, then reducing the height of one piece does not yield a cost improvement (and may even increase cost) as the overall build height of the bin remains the same; an appreciable cost improvement can only be gained once the height of all the pieces in the bin is reduced. For this reason, a slight temporary increase in cost (e.g. $\leq 1\%$) is permitted during the SHR stage, as long as the build height of the current bin continues to decrease. If no further height reductions are feasible in the current bin and the cost has not been improved, the current solution reverts back to the best solution in the search.

3.2. Single Bin Support Structure Volume Reduction (SSVR)

SSVR follows a very similar procedure to SHR; the only differences being the fitness used in the Tabu search, where height is replaced with support volume structure, and the iteration of pieces in P_i. Because the build height of b_i is determined by the tallest part in P_i, we sort P_i by non-decreasing height after each update and only take the first piece p_1 at each iteration. In contrast, the support structure volume of each piece can be reduced independently of other pieces. Hence, we sort P_i only once, in order of non-decreasing support structure volume, and iterate through each piece p_k, until $k = n$, where n is the length of P_i. Similarly to SHR, to fully explore the current solution neighbourhood, the SSVR stage allows a slight cost increase in the current solution if the support structure volume is reduced, but reverts back to the best solution if no cost improvement is achieved before moving on to the next bin.

3.3. Pairwise Bin Build Height Reduction (PHR)

In addition to changing the build orientations of pieces, we can also improve the bin assignment. For this reason the following two stages, PHR and PSVR, address bins in pairs. As before, the former addresses the height and the latter addresses the support structure volume of pieces. Unlike SHR and SSVR, PHR and PSVR address every possible pair of bins regardless of their utilisation. By pairing strong bins with weaker ones and by allowing the pieces to move between them, we hope to further utilise empty spaces within the weak bins and increase the freedom of movement inside strong bins.

As shown in Figure 3, the first step in PHR sorts the current set of bins B by non-increasing build cost. The second step takes a set of pieces $P_{ij} = P_i \cup P_j$ from a pair of bins b_i and b_j in B, where $i < j$. In the third step we order P_{ij} by non-increasing height and the fourth step performs a Tabu search to minimise the height of the tallest piece p_1. If the Tabu search is successful, the new piece p_1^* is accepted, so that $p_1 \leftarrow p_1^*$ in P_{ij}, and the final step in PHR solves a 2DIBPP for P_{ij}. As in other stages, if any bin $b \in B \setminus b_i, b_j$ contains a duplicate of p_1, the list of pieces in that bin must also be updated and repacked. If any of the bin repacking steps are unsuccessful, the solution is rejected.

Finally, if P_{ij} is successfully packed into a maximum of two bins b_i^* and b_j^*, where the combined cost of b_i^* and b_j^* is lower than the cost of b_i and b_j, the new solution is accepted and B is updated. Following this, P_{ij} is sorted by non-increasing height again and the process repeats until no further cost improvement can be made to b_i and b_j; at which point PHR repeats the above steps for the next pair of bins. PHR terminates when all pairs in B have been permuted.

3.4. Pairwise Bin Support Structure Volume Reduction (PSVR)

PSVR closely mirrors the steps in PHR, substituting support structure volume for height as the Tabu search objective. The main difference between these two stages is that PSVR sorts P_{ij} by non-increasing support structure volume, once for every pair of bins, then performs a Tabu search to minimise the support structure volume of every piece $p_k \in P_{ij}$, as opposed to only the tallest piece p_1 as done in PHR. The reason for this is explained in Section 3.2. After every successful Tabu search, the updated P_{ij} is repacked into a maximum of two bins, and each bin $b \in B \setminus b_i, b_j$ which contains the new piece is updated and repacked. If the repacked solution is both feasible and lower in cost than the current solution, this solution is accepted as the current solution and B is updated. The process is repeated for each pair of bins in B and terminates when all pairs have been permuted.

3.5. Bin Addition and Random Reassignment (BARR)

Each of the four stages described above tries to reduce the cost of the solution without increasing the overall number of bins. In this final stage of the ITSP, we explore the possibility of reducing the overall cost of the solution at the expense of adding an extra bin.

Following the improvement of individual bins in SHR and SSVR, and the bin assignment improvement in PHR and PSVR, it is going to be increasingly difficult for the procedure to find further feasible movements in the solution. Thus, by adding a new empty bin b_{M+1} and reassigning pieces from the original bins in B to b_{M+1}, we create free space in the disrupted bins which can be utilised to further improve the build orientations of remaining pieces. In this stage it is unclear which piece is likely to yield the biggest improvement in the overall cost. For this reason, in each iteration we select a random piece for reassignment and apply the Tabu search to all remaining pieces in the current bin before repacking. We also select bins randomly to further explore the solution space.

Thus, let b_R denote a random bin selected from B and p_r denote a random piece from the set of pieces P_R. To avoid re-visiting previously attempted pieces, p_r is stored in a 'Tabu' list (which is unrelated to the actual Tabu search). If p_r is placed into b_{M+1} successfully, the overall cost of the remaining set of pieces in b_R is minimised using the Tabu search. This is done iteratively for each piece in $P_R \setminus \{p_r\}$, using the change in cost to drive the search. For clarity, the details of this search are explained separately in Section 5.

If the Tabu search is successful, the next step is to repack the updated pieces into b_R. If the packing is unsuccessful, the piece with the smallest reduction in cost is reverted back to its original build orientation and the repacking is attempted again. This process is repeated until all pieces are successfully packed or until all pieces have been reverted to their original state. In the latter case, the solution yields no cost improvement and is therefore rejected. The procedure terminates either when a maximum number of no-improvement iterations is reached or when b_{M+1} is full – whichever occurs first.

4. Generating Build Orientations

Each geometry in G is provided in the form of a Stereolithography (STL) file, which is the industry-standard file format used to interface between 3D geometry data and AM machines. The STL file represents only the surface of the 3D geometry, defined by a set of triangular facets. Each facet is defined by a set of three vertices and a unit normal vector (pointing outward from the solid geometry). Each vertex and unit vector is defined in 3D Cartesian coordinates.

Hence, let a geometry g be defined by set of facets F, and a set of vertices V. Let each facet $f \in F$ be defined as $f(v_1, v_2, v_3, \hat{n})$, where v_1,v_2 and v_3 are three unique neighbouring vertices in V and $\hat{n}$ is the unit normal vector of f. Using this definition it is easy to find the height of g, by sorting the vertices in V by non-increasing z-coordinate and taking the difference of z_{max} and z_{min}.

To generate a piece p_g, a 2D polygon is created in the following way. The set of vertices V is projected onto the horizontal $x-y$-plane, such that $v_{3D}(x, y, z) \leftarrow v_{2D}(x, y)$, and a Delaunay triangulation is generated for the list of resulting 2D points. Each edge in the resulting mesh can either be defined as an internal edge, if it is shared by two triangles, or an outer edge, if the edge belongs to only one triangle. The convex polygon is defined by the set of unique vertices which belong to these outer edges. To produce a concave hull we use a 'digging' procedure similar to that of Duckham et al. (2008), where each large outer edge (e.g. length exceeding 20 mm) is removed from the polygon and the two exposed inner edges are added in its place.

Finally, the support structure volume must be estimated. This is done by calculating the area underneath any facet that exceeds the 45^{o} overhang limit, as explained in Section 1.1. To

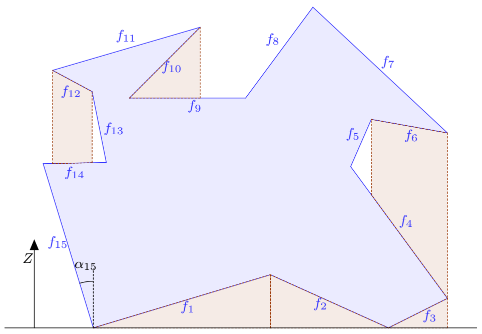


Figure 5: Schematic 2D example of support structure (red) for an arbitrary geometry (blue). Support structure is projected underneath each overhanging facet in F which exceeds 45^o with respect to the z axis, as indicated by α_{15} for f_{15}.

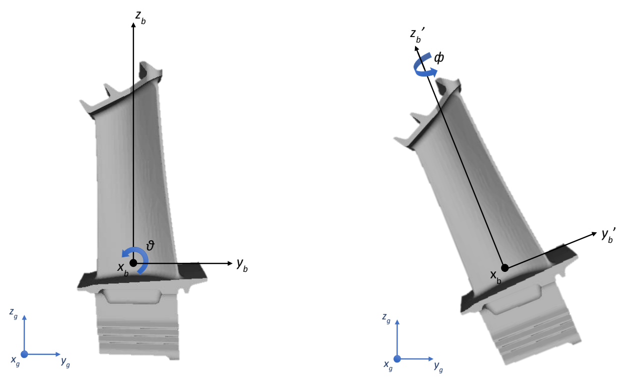
find the angle of facet f we can use the dot product of its normal vector $\hat{n}$ with the global z-axis. To estimate the support structure volume accurately, intersection of support structure with the geometry must be considered, as shown in Figure 5.

To check for support structure intersection, a series of facet-ray intersection checks are performed for each supported facet f – similar to the method used by Schwerdt et al. (2000). The ray is projected along the $< 0, 0, -1 >$ direction from f, and the check is performed for each upward-facing facet which lies beneath f. This intersection check is the most computationally expensive part of generating build orientations, with time complexity $O(nm)$, where n is the number of supported facets and m is the number of facets which lie beneath each supported facet.

When generating build orientations, two types of approaches have been proposed in literature: continuous rotation (Canellidis et al., 2006; Padhye & Deb, 2011; Peng et al., 2017) and discrete (Zhang et al., 2016b,a; Byun & Lee, 2006). One common discrete method, is to generate a 3D convex polygon of the geometry and consider each facet of the polygon as a potential base of the geometry (i.e. each facet corresponds to a candidate build orientation). Another way is to isolate specific design features or surfaces of the part, and generate the most favourable build orientation for each surface or feature. The former method is quite restrictive and poses the risk of missing optimal build orientations; while the latter is difficult to generalise across a large number of different parts. Yet another method, proposed by Zhang et al. (2016b), uses k-means clustering of surface facets to generate candidate orientations corresponding to the averaged normal vector of each cluster, however, this method requires additional pre-processing and careful selection of the k parameter. In order to avoid over-restricting the solution space early on in the problem, we allow continuous rotation in the build orientation search.

It is possible to define any build orientation by two consecutive rotations, using the orthogonal body-centric axes of g. Each build orientation is therefore denoted by $o(\theta, \phi)$, where θ and ϕ are two angles of rotation, as shown in Figure 6, where $0^o \leq \theta \leq 180^o$ and $0^o \leq \phi < 360^o$. Body-centric axes have been used to minimise the occurrence of duplicate build orientations. It should be noted at this point, that all values of ϕ will result in equivalent build orientations when $\theta = 0^o, 180^o$. If one of these build orientations happens to be a local minimum the Tabu search is likely to get stuck. To deal with this issue such build orientations are denoted by $o(0^o,^*)$ and $o(180^o,^*)$ in the Tabu list, respectively, where * represents any arbitrary value of ϕ.

The downside of continuous rotation is the vast solution space and the danger of an inefficient search, which may exceed the computational time budget before it can find a good build orientation. To tackle this, we limit movements to steps of one degree, since movements below



(a) Rotate anti-clockwise by θ about x_b-axis.

(b) Rotate anti-clockwise by ϕ about z_b-axis.

Figure 6: An example of the build orientation rotation procedure for a turbine blade. Each build orientation is generated by applying step (a), followed by step (b) to the geometry. Body-centric coordinates are denoted by x_b, y_b and z_b; global coordinates (x_g, y_g, z_g) are shown for reference.

this limit are likely to cause negligible changes to the Tabu search objective values. Additionally, we experiment with different step sizes in the Tabu search, as described in Section 6.

5. Local Tabu Search

Since the ITSP aims to explore a large number of build orientations it must do so in a relatively low number of iterations. Moreover, affected pieces must be repacked each time a build orientation is modified, which requires a computationally expensive local bin packing step; thus, the build orientation local search must be greedy enough to make this expensive step worthwhile. Tabu search (TS) is a well-known meta-heuristic used to guide a local search procedure and aid it in exploring the solution space beyond local minima. In addition to its simplicity and its ability to escape local minima, the convergence rate and greediness of TS can be controlled effectively through the size of the Tabu list, making it a suitable meta-heuristic for exploring different build orientations in the ITSP.

As outlined above, the ITSP uses Tabu search to find build orientation with different objectives, depending on the stage in the procedure:

- The first stage minimises piece area;
- SHR and PHR minimise piece height;
- SSVR and PSVR minimise support structure volume;
- BARR maximises the cost reduction in the bin.

Algorithm 1 shows the Tabu search procedure used for minimising the area a_p of piece p. The same procedure is used to minimise piece height h_p and support structure volume s_p, by substituting h_p or s_p for a_p, respectively. The termination condition of the algorithm is the maximum number of consecutive no-improvement iterations, denoted by N, and the size of the Tabu list, denoted by l. The neighbourhood construction at each iteration follows a simple greedy procedure, as shown in Figure 7. The shape of the neighbourhood is defined by the four lateral directions of movement, i.e. $\theta \pm d$ and $\phi \pm d$, as well as all possible combinations of these movements, resulting in a total of eight neighbouring solutions at every iteration; this

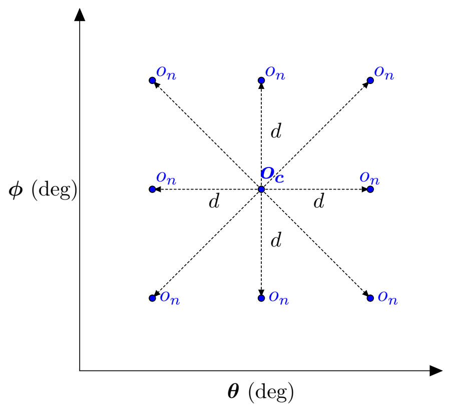


Figure 7: Local search neighbourhood structure, where o_c denotes the current build orientation, o_n denotes the neighbours and d denotes the step size; θ and ϕ are two rotation angles.

structure was chosen to ensure fast iterations in the Tabu search. The size of the neighbourhood is controlled by the step size, denoted by d.

To reduce the overall cost of bin b_i, the Tabu search must consider the set of bin pieces P_i collectively. Prior to the search, the pieces in P_i are sorted by non-increasing height. Starting from the tallest piece, the algorithm consecutively searches for a build orientation for each piece in P_i, using the cost difference at each iteration to drive the search. To obtain the change in bin cost, we use Equation 4, which follows from the objective function described in Section 2.

$$\triangle c = C_1 * \triangle h + C_2 * \triangle s \tag{4}$$

where $\triangle h$ is the change in build height and $\triangle s$ is the change in support structure volume, and C_{w_1} and C_{w_2} are their respective cost coefficients. Negative values of $\triangle c$ indicate a cost improvement, while positive values indicate a cost increase.

```
Input: $o_{in}, d, l, N$
Result: $o_b$
$o_c \leftarrow o_{in}$;
$o_b \leftarrow o_c$;
$n \leftarrow \varnothing$;
$TabuList \leftarrow \emptyset$;
while $n < N$ do
    Construct neighbourhood for $o_c$ using $d$ step size;
    Sort neighbours by non-decreasing piece area and return best neighbour $o_{n_1}$;
    if $TabuList.Size > l$ then
        Remove oldest entry in $TabuList$;
    end
    Add $o_c$ to $TabuList$;
    Set $o_c \leftarrow o_{n_1}$;
    if $a_b > a_c$ then
        $n \leftarrow \varnothing$;
        $o_b \leftarrow o_c$;
    end
    else
        $n \leftarrow n + 1$;
    end
end
```

Algorithm 1: Tabu search procedure, where o_{in}, o_c and o_b denote the initial, current and best build orientation, respectively; and a_b and a_c denote the areas of the best and current pieces, respectively.

The change in support structure volume is simply the difference between two Tabu search iterations of the same piece, in other words $\triangle s = s_{i+1} - s_i$, where i is the iteration counter. In the case of $\triangle h$, the difference must be taken between the tallest piece in P at iteration i, and the tallest piece at iteration $i + 1$. Thus, all pieces in P must be updated and sorted by non-increasing height after each iteration of the Tabu search. Additionally, in this version of the Tabu search, we constrain the area increase for each piece. The maximum percentage increase is denoted by A_{max}. The updated pseudo-code is shown in Algorithm 2.

```
Input: $o_{in}, d, l, N, A_{max}$, $P$
Result: $o_b$
$o_c \leftarrow o_{in}$;
$o_b \leftarrow o_c$;
$TabuList \leftarrow \emptyset$;
$n \leftarrow \varnothing$;
Sort pieces in $P$ by non-increasing height;
Set current bin height $h_b$ to height of $p_1$ in $P$;
Set initial area $a_1$ to area of piece $o_{in}$;
while $n < N$ do
    if $TabuList.Size > l$ then
        Remove oldest entry in $TabuList$;
    end
    Add $o_c$ to $TabuList$;
    Construct neighbourhood for $o_c$ using $d$ step size;
    foreach $o_n$ in $Neighbourhood$ do
        Set $P_n \leftarrow P$;
        Replace current piece $o_c$ in $P_n$ with piece $o_n$;
        Sort $P_n$ by non-increasing height;
        Set $h_n$ to height of $p_1$ in $P_n$;
        Set $s_n$ to $s$ of piece $o_n$;
        Set $s_c$ to $s$ of piece $o_c$;
        Set $\triangle h \leftarrow h_n - h_b$;
        Set $\triangle s \leftarrow s_n - s_c$;
        Set $\triangle c$ for $o_n$;
    end
    Sort solutions in neighbourhood by non-decreasing $\triangle c$;
    Return best neighbour $o_{n_1}$ with minimum $\triangle c$;
    Set $o_c \leftarrow o_{n_1}$;
    if $\frac{a_c - a_1}{a_1} \leq A_{max}$ then
        if $\triangle c < 0$ then
            $n \leftarrow \varnothing$;
            $o_b \leftarrow o_c$;
            Update current piece $o_c$ in $P$ and sort by non-increasing height;
            Set $h_b$ to height of $p_1$ in $P$;
        end
        else
            $n \leftarrow n + 1$;
        end
    end
    else
        Terminate search;
    end
end
```

Algorithm 2: Tabu search used in the BARR stage. Initial, best and current build orientations are denoted by o_{in}, o_b and o_c, respectively, while neighbour build orientations are denoted by o_n. P denotes the set of pieces inside the current bin; h denotes piece height and s denotes piece support structure volume. The neighbourhood step size and length of the Tabu list are denoted by d and l, respectively.

6. Implementation

The ITSP is coded in C# using the Visual Studio 2012 development environment. To solve the Mixed Integer Problem for the 2DIBPP we use Gurobi version 6.5.2. The code is parallelised on four i7-3820 cores, with 2.70 GHz maximum frequency.

To ensure good performance of the ITSP, a number of parameters needed to be tuned.

Firstly, a utilisation of 0.85 was considered to be the threshold between strong and weak bins. Secondly, the acceptance condition for each new build orientation was set to a minimum reduction of 1 mm when improving build height (i.e. 50 layers), and minimum reduction of 100 mm^3 for support structure volume (i.e. just under a gram of scrap material). Movements below these thresholds were not considered worthwhile since they would yield negligible cost improvements. In the SHR and SSVR stages, we allow some cost-increasing solutions, in the hopes of finding a better solution by delaying convergence. We limit the maximum allowable cost increase of such solutions to 1%. Finally, we constrain the maximum piece area increase in the Tabu search during the BARR stage to 30%. This value was selected after constraint values of 15% and 20% proved to be too stringent in the preliminary results.

The Tabu search parameters are perhaps the most crucial in determining the success of this procedure. Namely, these are the size of the Tabu list; the step size in each local neighbourhood; the maximum area constraint where applicable; and the number of no-improvement iterations before termination.

6.1. Tabu Search Parameters

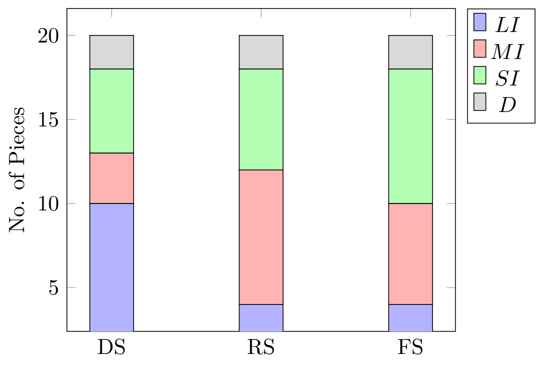
The idea behind the simplistic neighbourhood structure of the Tabu search (Figure 7) is to find a 'good enough' build orientation quickly, rather than attempting a more prolonged search for a better build orientation, which may ultimately lead to an infeasible bin solution. The same reasoning was used in selecting the step size and termination condition of the Tabu search.

We tested the following three strategies: Decreasing Step (DS), Random Step (RS) and Fixed Step (FS). At each iteration of the Tabu search, RS sets the step size to a random integer value between 1^o and 30^o, while FS keeps the step size fixed at 5^o throughout the search. The DS strategy starts with an initial step of 15^o, which is then multiplied by a reduction factor of 0.85 at each iteration. In all cases the search is terminated after eight iterations without improvement and the Tabu list stores 10 most recent build orientations.

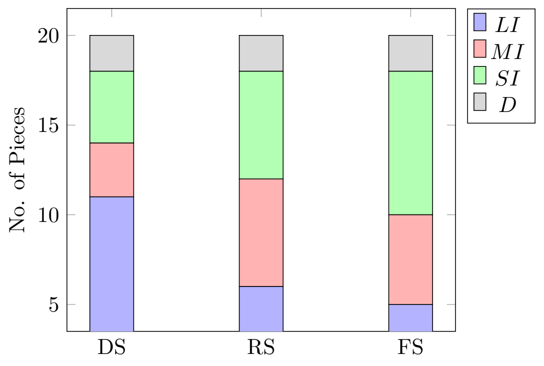
Test Pieces	Objective Improvement (%)			No. of Iterations			Solution Time (s)		
	DS	RS	FS	DS	RS	FS	DS	RS	FS
Aero housing 1	11.8	11.8	12.7	12	20	12	72	93	74
Aero housing 2	27.4	28.4	14.4	26	25	12	57	27	27
Alcoa bracket	41.2	52.4	48.1	11	24	32	37	61	107
Bearing block	52.9	51.2	51.5	32	14	12	24	15	10
Combustor plate 1	45.6	41.2	41.8	13	19	15	23	60	25
Combustor plate 2	85.8	84.4	84.4	11	13	11	312	326	340
Control arm	56.8	58.9	62.8	11	17	12	5	7	6
Engine block	0.9	0	1.3	16	8	9	4	2	2
GE bracket	26.5	21.6	20.0	25	16	26	378	458	441
Gear	0	0	0	8	8	8	0.4	0.6	0.4
Impeller	71.0	69.0	69.2	14	11	19	4,989	7,900	6,325
RC Jet bracket	34.7	32.7	31.6	18	19	28	1,996	3,325	3,112
RC Jet NGV ring	53.6	52.6	49.0	10	10	12	92	193	69
Seal segment	70.6	76.7	72.3	10	37	12	69	148	84
Sector	82.5	80.0	79.5	10	11	19	11	48	19
Swivel hinge	0	0	0	8	8	8	2	2	2
Turbine blade	40.7	40.8	40.2	12	20	19	245	607	367
Turbine wheel (Baumers)	0	0	0.6	8	8	10	23	41	31
Turbine wheel (turbocharger)	67.5	66.3	28.7	26	15	14	264	456	173
ULA bracket	50.2	39.7	13.4	20	18	25	105	105	116
Average	**41.0**	**40.4**	**36.1**	**15.5**	**17.2**	**15.7**	**576**	**901**	**759**

Table 1: Results for DS, RS and FS tested on 20 pieces.

The rationale behind the DS strategy is to quickly cover the solution space with a relatively large initial step and compare a number of potential local minima, before converging on the best local minimum with a decaying step size. The step size is kept constant for the first three iterations before applying the reduction factor. This prevents the search from converging prematurely if the initial solution happens to fall in a local minimum.



(a) Relative objective improvement for DS, RS and FS.



(b) Relative solution time for DS, RS and FS.

Figure 8: DS, RS and FS ranked by objective improvement (a) and number of iterations to final solution (b).

The values are selected based on empirical observations; 15° is a sufficiently large step to explore a significant area of the solution space without missing out too many solutions, while the selected reduction factor, which converges at 1° by the 18th iteration, yields an appropriate reduction gradient, given the average solution is reached after 16 iterations, as shown in Table 1.

The three strategies were tested on 20 different pieces with the objective of reducing the area of each piece. Table 1 presents the results using three measures of performance: objective improvement, which is the difference in piece area between the initial and best solutions; number of iterations before termination; and the total solution time for each piece. Both DS and RS outperform FS in terms of objective improvement, while FS and DS outperform RS in terms of iterations and solution time.

In general, DS outperforms both RS and FS for the majority of tested pieces. This is demonstrated by the two graphs shown in Figure 8. For each tested piece, the three strategies were ranked as 'Largest Improvement' (LI), 'Medium Improvement' (MI) and 'Smallest Improvement' (SI) based on the objective improvement, as shown in Figure 8a; and similarly ranked based on the number of iterations to final solution, as shown in Figure 8b. Two out of the 20 tested pieces ('Gear' and 'Swivel hinge') were ranked as 'Draw' (D), as all three strategies yielded no improvement and terminated after eight iterations. Based on these results, we employ the DS strategy in the Tabu search throughout the ITSP.

6.2. Test Data

In our tests the bin width W and length L are both set to 245 mm, while the bin height H is 275 mm. The no-build zones are approximated by four identical 20 mm squares in each corner of the bin, and each piece was enlarged by an even offset of 2.5 mm, to ensure a minimum separation of 5 mm between all pieces. In the build cost model we use an indirect cost rate of 26.64 £/h (taken from Baumers et al. (2016)) and base the material cost calculation on Cobalt-Chromium powder, a high-performance alloy commonly used in gas turbine applications, with a density of $8.3\,\mathrm{g\,cm^{-3}}$ and cost rate of $237.95\,£/\mathrm{kg}^3$. In the build time model, we use the regression model shown in Appendix A with input coefficients, r_1, r_2, r_3 and r_4, set to 0.5 h, $1.16 \times 10^{-1}\,\mathrm{h\,mm^{-1}}$, $2.04 \times 10^{-4}\,\mathrm{h\,mm^{-3}}$ and $8.33 \times 10^{-5}\,\mathrm{h\,mm^{-3}}$, respectively.

To test the ITSP, a total of 27 instances have been generated manually using the geometries listed in Appendix B. Out of 68 geometries, 24 were taken from literature, 41 were taken from GrabCAD, an online open-source community for sharing 3D geometry files, and three geometries were provided by Rolls-Royce plc. Additionally, three of the components taken from GrabCAD were specifically designed for SLM and were produced by three real-case design challenges sponsored by Alcoa, General Electric (GE) and United Launch Alliance (ULA), respectively. The generated instances are shown in Table 2.

Instance	N	D	Avrg. Vol. (mm^3)	Vol. Std. Dev. (mm^3)	Avrg. VR	Avrg. SA (mm^2)	Avrg. MLR	Avrg. no. of Facets
SI1	10	0	37,078	32,232	21,870	0.19	0.22	46,893
SI2	10	0	60,412	81,256	22,665	0.24	0.28	27,198
MI1	25	0	35,551	35,232	21,372	0.19	0.22	55,921
MI2	25	0	42,980	56,650	24,099	0.20	0.27	42,290
LI1	40	0	32,687	49,707	21,257	0.21	0.23	56,657
LI2	40	0	38,652	48,822	20,992	0.20	0.25	51,622
LAV1	25	0	4,791	3,912	7,413	0.22	0.13	39,344
LAV2	25	5	4,794	3,819	7,686	0.22	0.12	41,084
LAV3	25	12	4,500	4,064	7,658	0.24	0.13	51,138
MAV1	25	0	20,290	8,205	15,822	0.22	0.24	56,218
MAV2	25	5	19,938	8,275	15,517	0.23	0.24	57,756
MMV3	25	12	20,475	6,932	15,528	0.21	0.23	56,311
HAV1	25	0	70,152	50,441	36,265	0.18	0.32	101,244
HAV2	25	5	70,709	49,632	36,877	0.18	0.29	97,053
HAV3	25	12	70,412	49,159	36,331	0.18	0.29	71,158
LSDV	25	3	17,283	6,073	15,735	0.21	0.23	35,338
MSDV	25	3	29,718	21,747	20,692	0.18	0.22	96,728
HSDV	25	4	47,536	65,326	20,718	0.25	0.20	54,903
LVR	25	4	27,746	26,368	28,078	0.07	0.28	71,167
MVR	25	4	39,306	34,097	20,413	0.16	0.23	115,934
HVR	25	4	57,222	90,786	16,928	0.29	0.20	51,093
LMLR	25	4	7,922	9,679	5,624	0.31	0.09	32,722
MMLR	25	4	34,837	27,383	20,585	0.14	0.22	104,884
HMLR	25	4	48,226	55,593	30,169	0.13	0.33	124,617
LSA	25	3	6,786	8,707	4,250	0.31	0.14	19,561
MSA	25	4	19,487	15,310	15,107	0.15	0.23	50,277
HSA	25	3	63,952	53,593	39,780	0.12	0.31	82,144

Table 2: ITSP test instances. N is the total number of geometries and D is the number of duplicate geometries. SA is the surface area, VR is the volume ratio and MLR is the maximum length ratio.

The first six instances in Table 2 vary the number of pieces – where 'SI', 'MI' and 'LI' are used to denote small, medium and large instances, respectively. Geometries were selected randomly from the table shown in Appendix B to generate two instances of each size: SI1 and SI2 containing 10 geometries; MI1 and MI2 containing 25 geometries; and LI1 and LI2 containing 40 geometries. To limit computational expense, the remaining instances are fixed at a size of 25 geometries; this is also deemed a reasonable value for a low volume on-demand production scenario.

Next, we generate instances containing geometries of low (LAV1), medium (MAV1) and high (HAV1) average volume. Based on the available 68 test geometries, these instances are designed to approximately correspond to 4,500 mm^3 for LAV1, 20,000 mm^3 for MAV1 and 70,000 mm^3 for HAV1. We also consider the presence of duplicate geometries at this point, and generate two more instances of each type, with 20% (LAV2, MAV2, HAV2) and 50% (LAV3, MAV3, HAV3) of the total geometries being replaced with duplicates, respectively, while keeping all other variables at approximately the same values. Realistically, a few duplicates can be expected in a typical batch of parts, hence, this value is set to 12-16% of the total geometries for all remaining instances.

In addition to the size of pieces, the distribution of piece size has an equally direct effect on the success of the packing solution. For this reason, instances with low (LSDV), medium (MSDV) and high (HSDV) standard deviation of volume are also presented in Table 2.

The LSA, MSA and HSA instances correspond to low, medium and high average surface area of geometries, respectively, which has been suggested to have a correlation with the build time of SLM (Rickenbacher et al., 2013; Zhang et al., 2015b).

Finally, we consider the concavity and compactness of geometries, which can determine how well the pieces might fit together inside the bin, as well as how much support structure may be required. To quantify these two properties, we introduce two additional parameters, namely, maximum length ratio (MLR) and volume ratio (VR). To measure MLR, we find the longest vector that fits inside the geometry and divide its magnitude by the longest vector which fits inside the machine build envelope (i.e. a diagonal line between the front-bottom-left corner and the back-top-right corner of the 245mm x 245mm x 275mm bin). The VR is a ratio of the geometry volume divided by the volume of its bounding box. Instances of low, medium and large values were generated for both MLR and VR resulting in LMLR, MMLR and HMLR and LVR, MVR and HVR, respectively.

7. Computational Results

7.1. Analysis of ITSP Strategies

To analyse the effectiveness of the ITSP we compare the cost improvement for each of the five stages in the procedure relative to the initial solution. For clarity, the cost improvement, denoted by CI, is defined by Equation 5 for the overall ITSP, and Equation 6 for each stage in the procedure.

$$CI = S_i - \frac{S_{min}}{S_i} \tag{5}$$

where S_i is the cost of the initial solution and S_{min} is the cost of the best solution at the end of the ITSP.

$$CI = S_i^* - \frac{S_{min}^*}{S_i} \tag{6}$$

where S_i is the cost of the initial solution in the ITSP; S_i^* is the cost at the beginning of each stage; and S_{min}^* is the cost of the best solution found during that stage. It should be noted that a negative value of CI indicates an increase in the overall solution cost (i.e. negative improvement), and vice versa. Table 3 shows the average, best and worst CI produced by each stage and the overall ITSP, for the 27 tested instances; while Figure 9 shows the distribution of CI for each stage. As indicated by Table 3, SHR yielded mixed results, producing the single largest improvement (25.1% for the LSA instance) out of all the stages, but failing to reduce the cost for more than half of the tested instances, two of which resulted in a slight cost increase of 0.2%. On the other hand, SSVR produced the best results on average, followed closely by PSVR, as can be seen in Figure 9. The PHR and BARR stages yielded the least cost improvement, averaging 0.4% and 1.5%, respectively, and resulted in no improvement for majority of instances.

	Average CI (%)	Best CI (%)	Worst CI (%)	Average T (s)
SHR	2.6	25.1	-0.2	994
SSVR	6.7	22.6	0.2	7,758
PHR	0.4	4.0	0.0	668
PSVR	4.9	22.1	0.0	10,910
BARR	1.5	8.9	0.0	5,750
ITSP	16.0	31.0	4.0	29,092

Table 3: Summary of computational results showing average, best and worst cost improvement and average solution time for each stage in the procedure, as well as the overall ITSP. Cost improvement and solution time are denoted by CI and T, respectively.

There are a number of possible reasons why SHR and PHR performed worse than SSVR and PSVR. Firstly, this could be a natural outcome of the objective function, which is driven by support structure volume both through build time and material cost, while the build height parameter only influences the former. Consequently, the weight of support structure is likely to

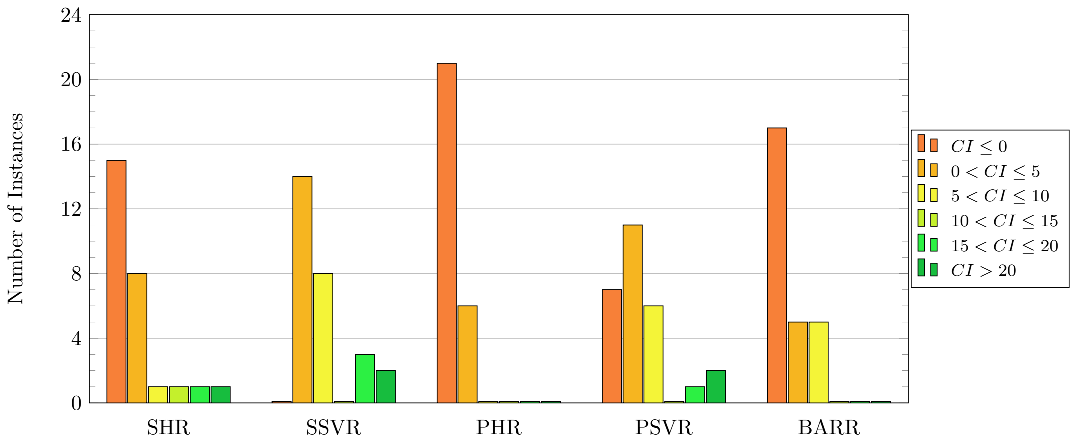


Figure 9: Histogram of cost improvement shown for each stage in the ITSP. Cost improvement is denoted by CI (%).

be higher than the weight of build height in the cost function, and since these two objectives are often conflicting, reducing build height may result in a higher overall cost, as in the case of the worst solution in Table 3. This is further supported by Figure 10, which compares the average variation in height (h_R), support structure volume (s_R), bin utilisation (u_R), and cost (c_R) during each stage in the ITSP. We observe a much flatter curve for h_R relative to s_R, with the cost reduction trend closely following that of s_R. In fact the h_R gradient shows a slight increase during SSVR and PSVR, indicating that cost improvements were gained by reducing support structure at the expense of build height.

It should also be noted that, to make the problem computationally feasible, we use a simplified model of support structure volume, which is likely to produce an overestimate. This is because we assume a fully dense support structure underneath all surfaces exceeding the self-supporting overhang limit; in reality, designers aim to minimise scrap by creating intelligent supports (e.g. lattice structures) tailored to the part. This discrepancy could be accounted for by scaling the support structure volume calculated by our model by some lattice density fraction, but to calculate its value some preliminary experimentation and model training would be required.

Another reason is that the SHR and PHR stages fail more often than SSVR and PSVR. This is partly due to the inverse correlation between the height and base area of the geometries; for example, when a tall prism with a small cross-section is moved from an upright position to resting on its side, the height of the piece decreases while piece area is increased. On the other hand, the relationship between support structure volume and piece area is weaker, since support structure is more dependent on the complexity and overhanging surface area of the geometry, than its aspect ratio. Furthermore, since build height can only be reduced by repositioning the tallest piece in the bin, SHR and PHR are only guaranteed as many iterations as there are bins (or pairs of bins) in the solution. Since most test instances in this paper were packed into a maximum of four bins, the SHR often failed after only four iterations, or less. In contrast, the SVR and PSVR stages iterate through every piece in the solution, regardless of the previous iteration – thus having greater chances of success. The upside of this is that both SHR and PHR terminated relatively quickly compared to SSVR and PSVR, as shown in Table 3.

Finally, the cost reduction yielded by SHR and PHR, is not dependent on the height of the tallest piece, but rather on the height difference between the two tallest pieces; this is in contrast to SSVR and PSVR, where pieces are independent of each other. This explains why SHR occasionally produces significant cost improvements for instances such as LSA, shown in

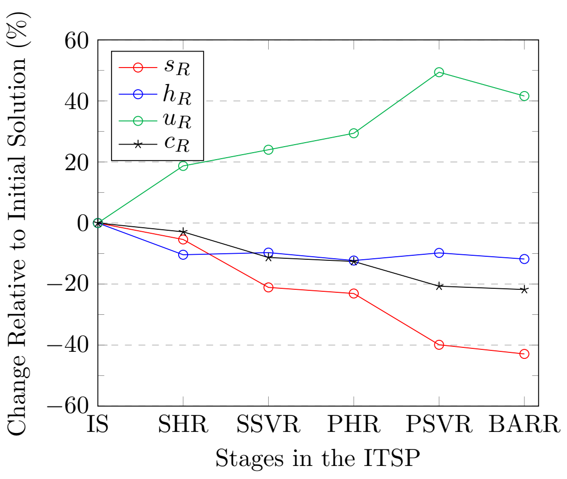
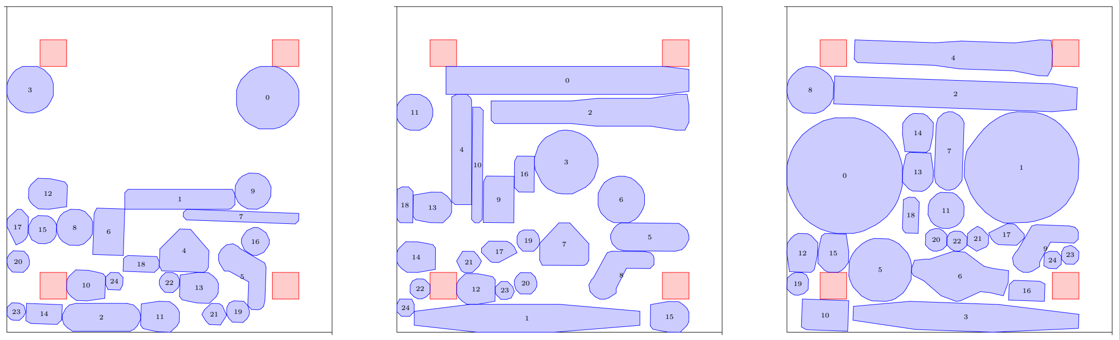


Figure 10: A comparison of the change in total support structure volume s_R, average build height of bins h_R, average bin utilisation u_R and total build cost c_R through the different stages of the ITSP, where IS stands for 'Initial Solution'. The values of s_R, h_R, u_R and c_R are calculated as a percentage difference between the initial solution and the best solution at the end of each stage, averaged across all test instances.

Figure 11, which consists of mostly small geometries and a few larger rod-shaped geometries, while failing to produce a large enough height decrease in most other instances.



(a) 1st iteration. $U = 0.14$, $H = 177.8$mm.

(b) 4th iteration. $U = 0.25$, $H = 82.3$mm.

(c) 7th iteration. $U = 0.41$, $H = 65.4$mm.

Figure 11: Solution steps for the LSA instance; (a) shows the initial solution, (b) shows an intermediate SHR solution and (c) show the final best solution produced by SHR. U and H denote bin utilisation (based on uninflated pieces) and build height, respectively.

Similarly, the poor results produced by PHR could be explained by a large increase in piece area after the Tabu search, and consequent failure of the 2DIBPP algorithm to find a feasible solution; Figure 10 shows that only a moderate increase in bin utilisation was achieved after PHR, compared to the subsequent PSVR stage.

As described in Section 3.3, PHR addresses bins in pairs in order to further utilise empty bin space in the solution. However, a successful iteration only guarantees a reduction in the height of the taller of the two bins; the height of the second bin is determined arbitrarily by the bin assignment during repacking. Thus, the PHR stage may be improved by considering the build height of the second bin as a constraint in the 1D bin packing model and the assignment mending strategy in the 2DIBPP algorithm. Alternatively, potential bin assignments could be ranked based on the standard deviation of height, encouraging pieces of similar height to be placed in the same bin, as proposed by Zhang et al. (2015a).

BARR is quite different to the other four stages in the ITSP. The purpose of this final stage in the procedure is to jump to an unexplored area in the solution space by adding an extra bin to the more-or-less converged solution at the end of PSVR. For 18 test cases no cost improvement was found and the number of bins remained the same, while in the other ten instances a cost reduction of up to 8.9% was gained by adding an extra bin, increasing the average number of bins from 2.2 to 2.6, as shown in Table 4. At this point it could be argued that in cases where BARR produced only a negligible cost improvement (e.g. $< 1\%$) the solution should be reverted to the best found solution at the end of PSVR, as the very small cost improvement does not justify the use of an additional bin. The best improvement (8.9%) produced by BARR was for the MMLR instance, which is shown in Figure 12 and Figure 13.

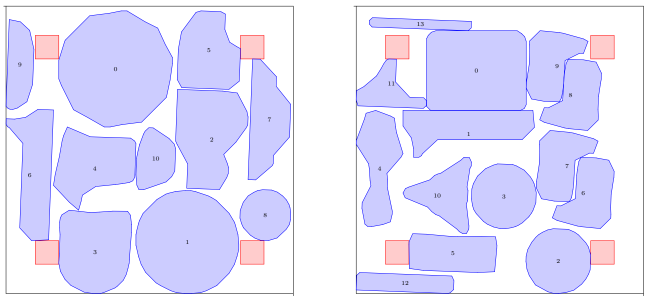

(a) Bin 1 (b) Bin 2

Figure 12: Solution for MMLR instance at the end of PSVR.

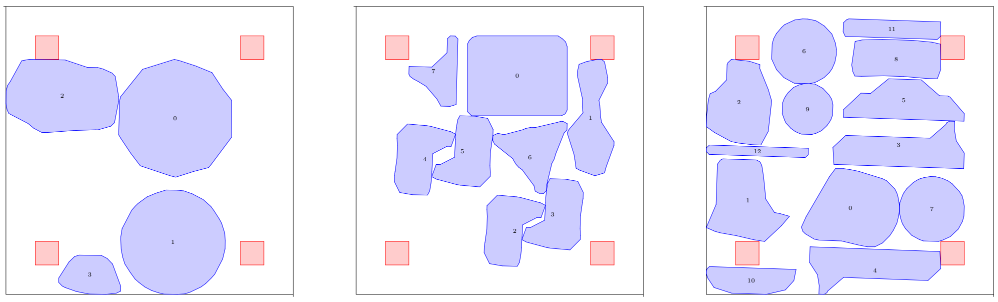

(a) Bin 1 (b) Bin 2 (c) Bin 3

Figure 13: Solution for MMLR instance at the end of BARR.

It should be noted that additional activities associated with the number of bins, such as machine set-up and part extraction, are not explicitly considered in the cost model of this paper, as they are difficult to measure and predict consistently. Such activities may have an impact on the effectiveness of the BARR stage, although it could be argued that this impact is negligible given the relatively high build times of current SLM machines. Moreover, the impact of bin-related costs would be further reduced for larger instances, as they would be absorbed across more bins.

7.2. Benchmarking

Due to the novelty of the problem, the ITSP procedure cannot be reproduced exactly by any currently existing method or software. To benchmark our results, we use a commercial software called Magics, which provides semi-automated decision support for process planning in ALM. As with other currently existing methods, Magics is only able to solve a simplified version of our problem, by addressing the build orientation and bin packing problems in two separate stages, independently of each other. Thus, we are only able to benchmark the initial solution of the ITSP, for which we use Magics version 20.03. The benchmark results were produced in the following two steps, keeping manual intervention to a minimum. Firstly, the build orientation of each geometry was optimised for minimum projected 2D area (i.e. piece area); secondly, all parts were placed into the minimum number of bins while keeping the build orientations fixed. As in the ITSP, a minimum distance of 5mm was maintained between all packed parts.

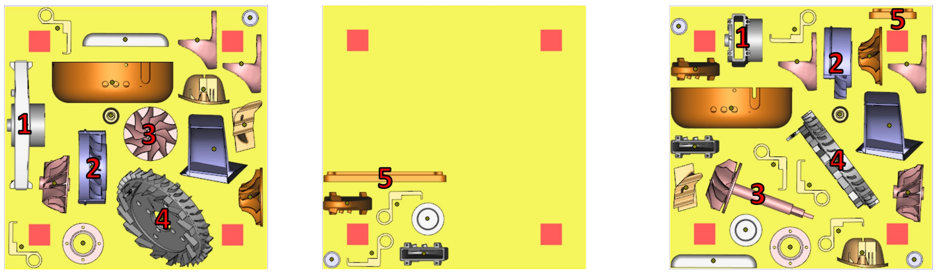
	Initial ITSP			Final ITSP			Magics			D_I (%)	D_F (%)
	C (£)	T (h)	M	C (£)	T (h)	M	C (£)	T (h)	M		
SI1	4,893.5	124.4	1	4,142.8	109.5	1	6,203.6	154.4	1	-21.1	-33.2
SI2	6,310.8	167.7	1	6,056.5	166.7	2	7,302.1	187.4	1	-13.6	-17.1
MI1	13,994.0	345.3	2	12,759.2	325.2	3	13,969.6	348.0	2	0.2	-8.7
MI2	15,434.2	386.6	2	13,718.7	357.0	2	14,407.6	366.6	2	7.1	-4.8
LI1	19,548.3	493.9	3	18,182.3	470.7	4	18,863.6	476.4	3	3.6	-3.6
LI2	21,481.3	541.9	3	19,092.2	493.6	3	22,005.2	554.5	3	-2.4	-13.2
LAV1	2,668.6	65.4	1	1,955.3	50.8	1	2,543.0	63.0	1	4.9	-23.1
LAV2	2,529.1	61.8	1	1,951.3	51.3	2	2,645.6	64.2	1	-4.4	-26.2
LAV3	2,798.1	71.4	2	2,116.6	56.5	2	3,090.3	73.5	1	-9.5	-31.5
MAV1	9,051.6	227.8	2	6,716.8	179.6	2	8,340.7	207.6	2	8.5	-19.5
MAV2	8,657.4	220.2	2	6,322.1	167.2	2	7,833.6	194.1	1	10.5	-19.3
MAV3	6,621.9	178.6	2	5,826.5	158.8	2	7,564.9	196.6	2	-12.5	-23.0
HAV1	25,595.6	648.0	4	22,293.0	587.7	5	24,289.6	614.2	3	5.4	-8.2
HAV2	25,561.2	643.9	4	22,922.1	590.9	4	25,209.6	633.3	3	1.4	-9.1
HAV3	25,525.3	639.1	4	22,382.9	577.8	4	22,908.0	577.7	3	11.4	-2.3
LSDV	7,303.2	186.1	2	5,501.0	147.9	2	7,234.9	177.7	1	0.9	-24.0
MSDV	10,474.9	267.3	2	9,468.3	249.2	3	10,134.1	260.7	2	3.4	-6.6
HSDV	16,730.5	419.6	2	14,794.0	387.3	3	16,647.6	420.4	2	0.5	-11.1
LVR	14,615.6	360.7	3	10,579.9	287.0	4	13,020.0	322.0	3	12.3	-18.7
MVR	13,376.5	338.4	2	11,479.7	304.2	3	12,190.7	315.1	2	9.7	-5.8
HVR	18,827.2	472.4	2	17,188.2	435.9	2	17,690.7	446.7	2	6.4	-2.8
LMLR	2,415.9	63.1	1	1,975.3	53.8	1	2,462.7	64.1	1	-1.9	-19.8
MMLR	13,039.3	320.2	2	10,927.3	284.6	3	11,735.6	295.0	2	11.1	-6.9
HMLR	17,551.5	443.0	3	16,412.9	420.5	4	16,609.6	418.4	3	5.7	-1.2
LSA	2,559.0	70.4	1	1,766.6	48.5	1	2,418.2	67.6	1	5.8	-26.9
MSA	7,746.6	195.7	2	5,971.2	159.2	2	7,426.6	189.8	2	4.3	-19.6
HSA	25,095.8	628.3	4	22,642.9	577.1	4	24,390.3	603.0	3	2.9	-7.2
Average	**12,607.7**	**317.8**	**2.2**	**10,931.3**	**285.1**	**2.6**	**12,190.3**	**307.1**	**2.0**	**1.9**	**-14.6**

Table 4: Comparison of ITSP initial and final solutions with Magics results. C denotes solution cost, M denotes the number of bins, H denotes average build height and SV denotes the total support structure volume.

Table 4 provides a comparison of Magics and ITSP results for the 27 test instances. It can be seen that the average initial cost of ITSP was marginally higher than the Magics benchmark, while the final ITSP solution outperformed the benchmark by an average of 14.6%. The difference between the benchmark and initial ITSP solution can be attributed to two factors; firstly, the support structure volume is higher than the benchmark by an average of 14%; secondly, seven of the 27 test cases produced one more bin than the benchmark, resulting in the average difference of 0.2 bins seen in Table 4.

Because Magics is a closed source software, we cannot provide a direct comparison of its bin packing and build orientation search methods to ours. However, we can make several suggestions and observations. As can be seen in Figure 14, the Tabu search was not able to find the minimum area solution for every piece, as in some cases it got stuck in a local minimum early on. Furthermore, in order to solve the problem in reasonable time, the Tabu search was allowed a maximum run time of 5 min per piece; as a result it was occasionally terminated early, usually in the case of large complex geometries (e.g. $\geq 500,000$ STL facets), such as piece 5 in Figure 14a. This explains why some ITSP initial solutions had more bins, and accounts for some of the difference in support structure volume.

It should be noted that, since there is no strong correlation between the support structure volume and piece area, the poor Tabu search solutions can sometimes work in favour of the ITSP, as in the case of SI1 and SI2, where the ITSP incidentally outperformed the benchmark



(a) ITSP solution (Bin 1). (b) ITSP solution (Bin 2). (c) Benchmark solution.

Figure 14: Comparisons of the benchmark and initial ITSP solution for the MAV2 instance. Pieces where the Tabu search performed worse than the benchmark are numbered in red.

due to lower support structure, while the number of bins remained the same due to the small size of the instances. Moreover, two build orientations with the same optimal piece area and height, can result in different volume of support structure, as shown in Figure 15. Although support structure and height are already used to break tie in the piece area Tabu search, the current heuristic is most likely to converge on whichever position is closest to the initial build orientation. The above shortcomings can be improved with further tuning of the Tabu search parameters.

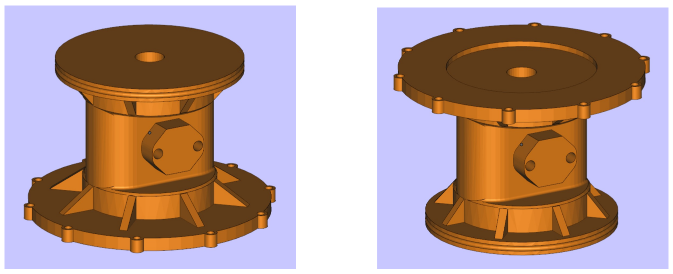

(a) Build orientation 1 (b) Build orientation 2

Figure 15: An example of two build orientations, which result in the same height, 60 mm, and piece area, 6699 mm^2, with support structure volume of 246,134 mm^3 and 105,645 mm^3 for (a) and (b), respectively.

The ITSP was also significantly slower, taking 10,979 seconds on average to process all STL files and produce an initial solution, compared to an estimated average solution time of 1,951 seconds for Magics, which also includes the loading of STL files and manual steps necessary to run the software. The overall solution time of the ITSP is 7.8 h on average for the 27 tested instances. As described in Section 4, the intersection checking performed in the support structure volume calculation, is by far the most time-consuming and computationally expensive aspect of the Tabu search, accounting for more than 80% of the time taken to generate each build orientation. Thus, improving the efficiency of this method or replacing it with an alternative, for example, a machine learning approach based on support structures designed by an expert user or Magics, could significantly improve the solution time of the ITSP (although other challenges would have to be considered, such as acquiring sufficient training data).

8. Conclusions and Future Research

In this paper we have addressed the combined build orientation and irregular bin packing problem in the context of SLM. This work was motivated by the HVLV production scenario which arises across different industries, such as aerospace and medical, and to which SLM is particularly suited since it is able to produce parts in mixed batches. Since our focus is on the aerospace sector, we have considered certification constraints in our procedure and tested it on a number of realistic aero-components provided by companies such as Rolls-Royce plc., General Electric and Alcoa. We developed a procedure consisting of six different stages and used total build cost as the objective. In each stage we used a Tabu search to solve the build orientation problem and a two-stage procedure to solve the 2DIBPP. The latter was solved by assigning pieces to potential bins, then solving the resulting 2D packing problem for each bin. Each stage explored a different area of the solution space by varying the bin assignment of pieces and by addressing a different aspect of the cost function in the Tabu search; namely, the build height, support structure volume and the number of bins in the solution.

To test the ITSP on a wide range of cases, we used a total of 68 geometries, some of which were taken from literature addressing similar problems, and generated 27 different test instances, varying the number, size and geometric properties of parts. The two most successful stages in the ITSP, SSVR and PSVR, were driven by reducing support structure volume in the Tabu search, and produced average cost improvements of 6.7% and 4.9%, respectively; while the build height-driven stages, SHR and PHR, produced average cost improvements of 2.6% and 0.4%, respectively. The BARR stage produced non-trivial improvements of 2.4-8.9% for seven of the 27 test cases. On average, the initial bin utilisation was increased by over 40% at the end of the ITSP (after a slight reduction in the BARR stage) indicating that the proposed procedure was quite successful in effectively utilising machine space. The relative success of the BARR stage suggests that increasing the number of bins can improve the cost, even for relatively small instances, provided that the additional machine space is utilised to improve the build orientation of parts. Additionally, using more bins allows the parts to be built in parallel; this presents an interesting trade-off between build cost, time and the number of machines, which could be addressed as part of future work.

Despite its long running time, the final ITSP solution yielded a cost improvement over the benchmark in all 27 test cases, with an average of 14.6% and ranging up to 33.2%. This demonstrates the advantage of solving the combined problem of build orientation, bin assignment and bin packing simultaneously to minimise build cost – a problem that has not been addressed previously.

To improve the accuracy, as well as the time, of this model, a machine learning algorithm could be used to predict the support structure volume. Since the vector-based support volume calculation is the most time-consuming part of the ITSP, it is expected that this modification could reduce the average running time of the procedure by as much as several hours, however, it would require some overhead model building prior to running the ITSP, and would need to be re-trained for new sets of geometries. The solution time could also be improved further by discretisation the build orientation solution space with a pre-processing method, such as the one proposed by Zhang et al. (2016b). However, it should be noted that limiting the number of candidate build orientations in this way would increase the risk of missing potentially very good solutions.

The work presented in this paper has opened up a number of new areas for further research. For instance, post-processing costs, such as surface polishing, and considerations of part dimensional accuracy and build failure due to residual stresses could be included into the objective function. The effects of the 2D orientation of pieces on the build quality of parts could also be explored.

Acknowledgements

This work is jointly funded by the Engineering and Physical Sciences Research Council (EPSRC) and Rolls-Royce plc. We would like to thank Martin Baumers and Vassilios Canellidis for providing test geometries which were used in this paper.

References

Ahn, D., Kim, H., & Lee, S. (2007). Fabrication direction optimization to minimize post-machining in layered manufacturing. *International Journal of Machine Tools and Manufacture*, *47*, 593–606.

Alexander, P., Allen, S., & Dutta, D. (1998). Part orientation and build cost determination in layered manufacturing. *CAD Computer Aided Design*, *30*, 343–356.

Araújo, L., Ozcan, E., Atkin, J., Baumers, M., Tuck, C., & Hague, R. (2015). Toward Better Build Volume Packing in Additive Manufacturing: Classification of Existing Problems and Benchmarks. *Proceedings of the Solid Freeform Fabrication Symposium*, (pp. 401–410).

Atzeni, E., & Salmi, A. (2012). Economics of additive manufacturing for end-usable metal parts. *The International Journal of Advanced Manufacturing Technology*, *62*, 1147–1155.

Baumers, M., Dickens, P., Tuck, C., & Hague, R. (2016). The cost of additive manufacturing: machine productivity, economies of scale and technology-push. *Technological Forecasting and Social Change*, *102*, 193–201.

Byun, H. S., & Lee, K. H. (2006). Determination of the optimal build direction for different rapid prototyping processes using multi-criterion decision making. *Robotics and Computer-Integrated Manufacturing*, *22*, 69–80.

Calignano, F. (2014). Design optimization of supports for overhanging structures in aluminum and titanium alloys by selective laser melting. *Materials & Design*, *64*, 203–213.

Canellidis, V., Dedoussis, V., Mantzouratos, N., & Sofianopoulou, S. (2006). Pre-processing methodology for optimizing stereolithography apparatus build performance. *Computers in Industry*, *57*, 424–436.

Canellidis, V., Giannatsis, J., & Dedoussis, V. (2009). Genetic-algorithm-based multi-objective optimization of the build orientation in stereolithography. *The International Journal of Advanced Manufacturing Technology*, *45*, 714–730.

Canellidis, V., Giannatsis, J., & Dedoussis, V. (2013). Efficient parts nesting schemes for improving stereolithography utilization. *CAD Computer Aided Design*, *45*, 875–886.

Chlebus, E., Kuznicka, B., Kurzynowski, T., & Dybala, B. (2011). Microstructure and mechanical behaviour of Ti6Al7Nb alloy produced by selective laser melting. *Materials Characterization*, *62*, 488–495.

Das, P., Chandran, R., Samant, R., & Anand, S. (2015). Optimum Part Build Orientation in Additive Manufacturing for Minimizing Part Errors and Support Structures. *Procedia Manufacturing*, *1*, 343–354.

Duckham, M., Kulik, L., Worboys, M., & Galton, A. (2008). Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern Recognition*, *41*, 3224–3236.

Frank, D., & Fadel, G. (1995). Expert system-based selection of the preferred direction of build for rapid prototyping processes. *Journal of Intelligent Manufacturing*, *6*, 339–345.

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

Ghobbar, A. (2004). Forecasting Intermittent Demand for Aircraft Spare Parts: A Comparative Evaluation of Methods. *Journal of Aircraft*, *41*, 665–673.

Gogate, a. S., & Pande, S. S. (2008). Intelligent layout planning for rapid prototyping. *International Journal of Production Research*, *46*, 5607–5631.

Guo, N., & Leu, M. C. (2013). Additive manufacturing: Technology, applications and research needs. *Frontiers of Mechanical Engineering*, *8*, 215–243.

Holmström, J., Partanen, J., Tuomi, J., & Walter, M. (2010). Rapid manufacturing in the spare parts supply chain. *Journal of Manufacturing Technology Management*, *21*, 687–697.

Hopkinson, N., & Dickens, P. M. (2003). Analysis of rapid manufacturing – using layer manufacturing processes for production. *Proceedings of the Institute of Mechanical Engineers, Part C : Journal of Mechanical Engineering Science*, *217*, 31–39.

Ikonen, I., Biles, W. E., Kumar, A., Ragade, R. K., & Wissel, J. C. (1997). A Genetic Algorithm for Packing Three-Dimensional Non-Convex Objects Having Cavities and Holes. *Icga*, (pp. 591–598).

Järvinen, J.-P., Matilainen, V., Li, X., Piili, H., Salminen, A., Mäkelä, I., & Nyrhilä, O. (2014). Characterization of Effect of Support Structures in Laser Additive Manufacturing of Stainless Steel. *Physics Procedia*, *56*, 72–81.

Jhabvala, J., Boillat, E., & Andre (2011). An innovative method to build support structures with a pulsed laser in the selective laser melting process. *The International Journal of Advanced Manufacturing Technology*, *59*, 137–142.

Khajavi, S. H., Partanen, J., & Holmström, J. (2014). Additive manufacturing in the spare parts supply chain. *Computers in Industry*, *65*, 50–63.

Lan, P.-T., Chou, S.-Y., Chen, L.-L., & Gemmill, D. (1997). Determining fabrication orientations for rapid prototyping with Stereolithography apparatus. *Computer-Aided Design*, *29*, 53–62.

Leary, M., Merli, L., Torti, F., Mazur, M., & Brandt, M. (2014). Optimal Topology for Additive Manufacture: A method for enabling additive manufacture of support-free optimal structures. *Materials & Design*, *63*, 678–690.

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

Padhye, N., & Deb, K. (2011). Multi-objective optimisation and multi-criteria decision making in SLS using evolutionary approaches. *Rapid Prototyping Journal*, *17*, 458–478.

Peng, A. H., Ghasri-khouzani, M., Gong, S., Attardo, R., Ostiguy, P., Gatrell, B. A., Budzinski, J., Tomonto, C., Neidig, J., Shankar, M. R., Billo, R., Go, D. B., & Hoelzle, D. (2017). Optimization of Build Orientation for Minimum Thermal Distortion in DMLS Metallic Additive Manufacturing. In *Solid Freeform Fabrication Symposium* 2 (pp. 820–835).

Piili, H., Happonen, A., Väistö, T., Venkataramanan, V., Partanen, J., & Salminen, A. (2015). Cost Estimation of Laser Additive Manufacturing of Stainless Steel. *Physics Procedia*, *78*, 388–396.

Rickenbacher, L., Spierings, a., & Wegener, K. (2013). An integrated cost-model for selective laser melting (SLM). *Rapid Prototyping Journal*, *19*, 208–214.

Ruffo, M., & Hague, R. (2007). Cost estimation for rapid manufacturing - simultaneous production of mixed components using laser sintering. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *221*, 1585–1591.

Saha, B., Wanhill, R. J. H., Eswara Prasad, N., Gouda, G., & Tamilmani, K. (2013). Airworthiness Certification of Metallic Materials. In *Aluminum-Lithium Alloys: Processing, Properties, and Applications* (pp. 537–554).

Schwerdt, J., Smid, M., Janardan, R., & Johnson, E. (2000). Protecting critical facets in layered manufacturing: Implementation and experimental results. *CAD Computer Aided Design*, *35*, 647–657.

Thomas, D. (2009). *The Development of Design Rules for Selective Laser Melting*. Ph.D. thesis Cardiff University.

Thomas, D. (2016). Costs, benefits, and adoption of additive manufacturing: a supply chain perspective. *The International Journal of Advanced Manufacturing Technology*, *85*, 1857–1876.

Wodziak, J. R., & Fadel, G. M. (1999). Packing and optimizing the center of gravity location using a genetic algorithm. In *Journal of Computers in Industry*.

Wu, S., Kay, M., King, R., Vila-Parrish, A., & Warsing, D. (2014). Multi-objective optimization of 3D packing problem in additive manufacturing. In *IIE Annual Conference and Expo 2014* (pp. 1485–1494). Institute of Industrial Engineers.

Zhang, Y., Bernard, A., Harik, R., & Fadel, G. (2017). A New Method for Single-Layer-Part Nesting in Additive Manufacturing. *Rapid Prototyping Journal*, .

Zhang, Y., Bernard, A., Harik, R., & Karunakaran, K. P. (2015a). Build orientation optimization for multi-part production in additive manufacturing. *Journal of Intelligent Manufacturing*, *28*, 1393–1407.

Zhang, Y., Bernard, A., Valenzuela, J. M., & Karunakaran, K. P. (2015b). Fast adaptive modeling method for build time estimation in Additive Manufacturing. *CIRP Journal of Manufacturing Science and Technology*, *10*, 49–60.

Zhang, Y., Gupta, R. K., & Bernard, A. (2016a). Two-dimensional placement optimization for multi-parts production in additive manufacturing. *Robotics and Computer-Integrated Manufacturing*, *38*, 102–117.

Zhang, Y., Harik, R., De Backer, W., & Bernard, A. (2016b). A Facet Cluster-Based Method for Alternative Build Orientation Generation in Additive Manufacturing. In *Solid Freeform Fabrication 2016: Proceedings of the 27th Annual International* (pp. 23–35).

Appendix A. Build Time Model

The data used to create the build time regression model are given in Table A.5. All six batches contained only identical parts and were built on an EOS M270 machine, using Cobalt Chromium MP1 powder and a constant layer thickness of 20 µm.

	N	V_P (mm^3)	V_S (mm^3)	H (mm)	T_R (h)	T_P (h)	T_S (h)
Batch 1	81	67,149	0	12.4	1.6	0	1.6
Batch 2	3	99,186	145,566	34.1	23.2	20.9	4.2
Batch 3	2	80,776	394,722	39.0	18.5	33.7	4.6
Batch 4	3	595,647	0	53.0	119.9	0	6.5
Batch 5	6	63,654	173,358	28.8	14.4	8.6	4.5
Batch 6	1	167,188	119,909	131.0	34.9	13.2	15.6

Table A.5: Regression model input data. N denotes the number of parts in the batch; T_R denotes total re-coating time; T_P denotes total part exposure time; T_S denotes total support structure exposure time; V_P and V_S denote total part volume and support structure volume, respectively; and H denotes batch build height.

A simple linear regression model of build time was produced in Microsoft Excel 2013, using part volume v, support structure volume s and build height h as model inputs. The model is summarised in Table A.6. The original value of r_1 provided by the model was -12.17 h; this was changed to 0.5 h to provide a more realistic model constant (especially since the data does not include factors such as preheating, cooling etc.) and to discourage the ITSP from increasing the number of bins in the solution (unless such an increase offers a significant cost improvement). The model maximum error of -32.5% and RMSE of 5.48 h are quite high, as to be expected for such a small set of training data. However, given the reasonable fit shown in Figure A.16, we consider it sufficient for the purpose of testing our methodology. This model can easily be replaced by the user with a more sophisticated one in the future.

Parameter	Value	Unit
r_1	0.5	h
r_2	1.16×10^{-1}	h mm^{-1}
r_3	2.04×10^{-4}	h mm^{-3}
r_4	8.33×10^{-5}	h mm^{-3}
R^2	0.98	
ME	-32.5	%
RMSE	5.48	h

Table A.6: Regression model coefficients and statistical measures, where ME and RMSE are the maximum and root mean square errors, respectively. r_1 denotes the constant coefficient, r_2 denotes the build height coefficient, r_3 denotes the part volume coefficient and r_4 denotes the support structure volume coefficient.

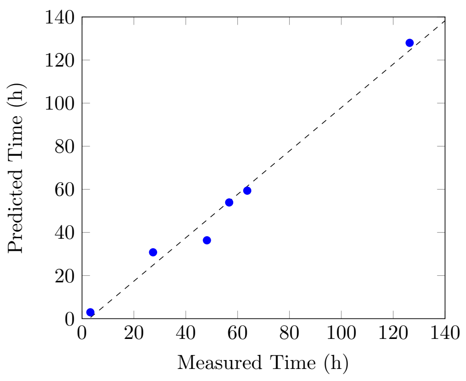


Figure A.16: Measured vs. predicted total build time for the six test batches.

Appendix B. Test Geometries

Group & Source	*Geometry Name*	*Volume (mm^3)*	*Surface Area (mm^2)*	*Volume Ratio (VR)*	*Max Length Ratio (MLR)*	*No. of Facets*
Araújo et al. (2015)	Bearing block	96,646	28,939	0.19	0.32	12,846
	Belt link	16,595	8,057	0.55	0.15	16,148
	End cap	1,766	2,217	0.23	0.09	26,892
	Turbine	20,618	11,625	0.25	0.13	107,806
	Venturi tube	960	1,042	0.40	0.07	59,294
Canellidis et al. (2006, 2009)	Caliper	51,601	23,483	0.17	0.21	43,154
	Card slot cover	11,759	14,710	0.09	0.23	12,498
	Cover	19,162	17,478	0.25	0.26	12,300
	Distributor	7,149	13,189	0.10	0.14	4,028
	Engine block	38,595	28,112	0.18	0.21	4,924
	Flip-top	2,073	4,427	0.07	0.15	6,552
	Gear	540	659	0.39	0.04	1,420
	Handle	28,250	29,844	0.08	0.27	9,698
	Impeller	80,661	40,349	0.17	0.28	474,222
	Insect trap	5,631	16,474	0.07	0.17	50,910
	Jawbone	73,270	22,190	0.08	0.31	100,948
	Phone cover	53,704	57,385	0.08	0.45	30,004
	Pipe support	2,296	2,959	0.24	0.09	620
	Sector	90,050	52,234	0.10	0.50	14,574
	Swivel hinge	10,952	7,647	0.22	0.15	4,860
	Washing machine arm (centre section)	28,552	29,441	0.14	0.30	17,718
	Washing machine arm (left section)	20,550	20,624	0.17	0.30	10,394
	Washing machine arm (right section)	20,551	20,624	0.17	0.30	10,370
Gearbox Assembly (GrabCAD)	Cover plate	27,134	13,880	0.38	0.28	19,972
	GB housing	282,507	69,052	0.28	0.42	66,658
	Offset shaft	31,520	8,798	0.78	0.42	6,596
	Plate lug	78,908	20,952	0.28	0.23	19,294
	Worm gear	37,698	14,996	0.39	0.15	16,262
	Worm gear shaft	23,995	7,243	0.26	0.34	5,502
RC Jet Assembly (GrabCAD)	Tube	19,870	16,184	0.14	0.25	247,318
	Axle	18,750	6,078	0.44	0.39	1,296
	Bearing	1,920	1,489	0.57	0.05	2,516
	Bracket	6,900	14,739	0.02	0.39	308,318
	Compressor bearing	2,770	2,504	0.32	0.07	34,424
	Compressor housing	48,970	30,367	0.11	0.27	77,780
	RC compressor wheel	8,760	11,461	0.13	0.13	15,640
	Diffuser	34,770	31,326	0.09	0.25	554,636
	Exhaust inner cone	3,280	9,492	0.05	0.11	44,816
	Exhaust nozzle	8,240	28,327	0.02	0.21	71,812
	Front bolt	880	710	0.47	0.04	1,852
	Front casing	11,960	40,358	0.02	0.27	127,916
	Jet Turbine bearing	305	435	0.42	0.03	1,374
	NGV ring	10,520	17,836	0.08	0.17	60,350
	Rear bolt	477	504	0.56	0.03	1,012
	Rear casing	57,230	82,533	0.04	0.35	170,028
	Spark plug	569	487	0.33	0.05	1,750
	Spring	767	2,131	0.16	0.06	192,536
	Starter	56,830	9,866	0.20	0.23	620,764
	Turbine casing	4,633	4,792	0.23	0.19	9,328
	RC turbine wheel	15,560	14,763	0.06	0.24	9,386
Turbocharger Assembly (GrabCAD)	Clamp	9,780	7,977	0.15	0.19	8,218
	Compressor flow	69,234	43,022	0.15	0.31	22,236
	TC compressor wheel	8,761	11,464	0.13	0.13	23,316
	TC housing	121,353	33,752	0.23	0.24	9,336
	Socket & cap screw	20,529	5,668	0.46	0.13	3,100
	Turbine flow	81,678	55,912	0.11	0.34	16,548
	TC turbine wheel	15,664	14,769	0.06	0.24	12,578
Rolls Royce	Combustor section 1	46,455	44,154	0.06	0.43	20,648
	Combustor section 2	64,000	48,494	0.15	0.53	8,186
	Seal segment	14,544	25,798	0.14	0.25	51,376
Aero-Parts (GrabCAD)	Aero-bearing bracket	40,806	20,037	0.12	0.30	32,686
	Control arm	90,512	33,471	0.13	0.42	7,654
	Fuel injector	8,077	7,644	0.13	0.16	39,272
	GE bracket	76,857	78,004	0.06	0.44	84,468
	Housing1	40,840	26,450	0.13	0.21	48,586
	Housing2	65,620	31,605	0.14	0.24	25,702
	Space bracket	30,540	13,682	0.14	0.25	75,982
	Turbine blade	13,216	10,478	0.11	0.22	31,230

Table B.7: Test geometries and their properties.