Optimal 3D printing of complex objects in a 5–axis printer

B. Ramos* D. Pinho† D. Martins* A.I.F. Vaz* L.N. Vicente‡

March 18, 2021

Abstract

Three-dimensional (3D) printing, also known as additive manufacturing (AM), has emerged in the last decades as an innovative technology to build complex structures. It enables increasing design complexity and low-cost customization with a vast range of materials. AM capabilities contributed to a widespread acceptance of 3D printing in different industries such as the aerospace and the automotive. However, important issues and limitations still need to be addressed, namely in printing complex objects where supports and material roughness surface are to be minimized.

In this work we consider a 5–axis printer with the three traditional xyz movements and two additional degrees of freedom on the printer table bed. These extra degrees of freedom (table bed rotation and tilt) allow printing of more complex objects, and we propose an approach which consists on the decomposition of complex objects into simpler parts, allowing each part to be printed in an optimal way. We aim to reduce the number of supports needed and attain high final object quality due to lower material surface roughness.

The optimal printing direction (or, equivalently, rotation) and sequencing of the object parts is determined by solving a combinatorial sequencing optimization problem. All the local or global optimal parts rotations are obtained by solving a global optimization sub-problem for each part, and are taken as input parameters for the sequencing optimization problem. We provide a heuristic approach for the combinatorial sequencing optimization problem, and a multistart multisplit search methodology for computing all the local or global optimal parts rotations for the sub-problems.

Keywords— additive manufacturing path planning, 3D printing, 5–axis printer, local and global optimization, optimal sequencing

1 Introduction

Three-dimensional (3D) printing also known as additive manufacturing (AM) has emerged in the last decades, becoming an alternative to the traditional subtractive manufacturing. AM builds up a component through the deposition of materials layer-by-layer, in opposition to starting with an over dimensioned raw block and removing unwanted material as in conventional subtractive manufacturing [8, 32]. It became a promising alternative for fabricating components made of expensive materials,

*Algoritmi Research Centre, University of Minho, 4710-057 Braga, Portugal, bruna.ramos@dps.uminho.pt, martinsdanielalopes@gmail.com, aivaz@dps.uminho.pt

†Algoritmi Research Centre, University of Minho, 4710-057 Braga and Research Centre in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253 Bragança, Portugal, diana@ipb.pt

‡Department of Industrial and Systems Engineering, Lehigh University, 200 West Packer Avenue, Bethlehem, PA 18015-1582, USA and Centre for Mathematics of the University of Coimbra (CMUC), lnv@lehigh.edu

thanks to its many benefits and affordable prices, leading to a growing revolution on AM at a global scale. From education to health, from archeology to engineering, 3D printers are already making significant practical impacts [4].

Most of the established 3D printing technologies are based on layered manufacturing, e.g., fused deposition modeling (FDM) or stereo lithography. Such a technology requires four main preparation tasks to be able to print a complete object [20]: i) object orientation – computation of the best orientation for the object to be built; ii) design of supports – to hold, during the printing process, the overhang parts of the object; iii) slicing – extracting layers (as sets of 2D polygons) from the object, converting the 3D object into 2D images; iv) path planning – extruder head path for printing. In particular, FDM printers build a sequence of structures by depositing material bottom-up layer-by-layer by heating and extruding thermoplastic filaments [25, 33, 35]. Typically, the extruder travels along the x and y axes to build an object layer and along the z axis to build the layers. Printers limited to movements in the 3-axis need to introduce support structures to support overhanging layers or overhangs higher than 45 degrees from the vertical axis [35]. An immediate consequence of this approach is that the effective printing resolution and consequently the resulting surface smoothness is anisotropic [33].

The type of printer considered herein provides the 3 traditional xyz movements together with two additional degrees of freedom on the printer table, allowing rotation and tilt (inclination) of the table. These extra degrees of freedom will allow printing of more complex objects with improvements in the surface quality and reduction of support structures. A 5–axis system enables re-orientation of the object during the printing process, being extremely useful for 3D printing, since overhanging structures can be minimized or removed.

There is extensive research literature on AM related fields like computational design for AM [10, 18, 19], AM processes [14, 18], process modeling and optimization [3, 7, 26, 31, 37], material science [18], and energy and sustainability [34]. However, additive manufacturing for 3D printing of complex objects by its decomposition into parts was only recently addressed. Let us briefly review these recent studies.

Ding *et al.* [8] addressed a new strategy for multi-direction slicing of CAD models in STL (Standard Triangle Language or Standard Tessellation Language) format by considering an optimal volume decomposition-regrouping strategy, applying a curvature-based volume decomposition method, which decomposes complex objects into sub-volumes using a depth-tree structure. Wang *et al.* [35], in order to improve the surface quality in 3D printing, presented a pipeline of algorithms that compute an object decomposition by using the co-compatibility of the facet normals with the printing directions. A 3D Voronoi diagram is computed to consolidated the parts shape. This technique has the particularity that the (manual) assembly order of parts is collision free, and parts ordering and direction for assembling are also obtained [36]. Massoni *et al.* [24] proposed a method that automatically decomposes 3D complex models into parts with the goal of lowering overall production costs. The proposed approach generates many alternative parts by using iterative cutting planes, followed by an exhaustive list of manufacturing plans for each assembly option where costs are estimated. A beam search optimization algorithm was applied with the search space organized as a decision tree, providing the best assembly and manufacturing cost. Luo *et al.* [22] proposed a framework called *chopper* also based on the beam search algorithm, decomposing large 3D objects into smaller parts so that each part fits into the printing volume space. Parts are assembled by the user to form the complete object.

We take advantage of the 5–axis printer in order to propose an approach where complex objects are decomposed into simpler parts, allowing each part to be printed in an optimal way, reducing the number of supports needed and attaining high final object quality due to the lower material surface roughness. Furthermore, the proposed approach builds complex objects without the user intervention to assembly the parts. Our approach relies on the solution of a master optimization problem for the optimal parts sequencing, where the optimal time sequence of parts to be built is determined. It consists of a nonlinear optimization problem with black-box type constraints over binary variables, and a heuristic is developed for its solution. The data defining this master problem requires the a priori calculation, for each part, of the printing directions (or orientations) that minimize the staircase effect (or alternatively the support area or building time). A bound constrained nonconvex optimization problem in continuous variables is posed for this purpose, one for each part. In fact, we are interested

in calculating as many local or global solutions of this latter problem as possible, as they correspond to different optimal printing directions. A multistart multisplit local search (MMLS) algorithm for continuous global optimization is proposed to identify as many local or global solutions as possible. The many optimal solutions computed by the MMLS algorithm are crucial to the sequencing heuristic, because more printing directions identified for each part lead to more feasible time sequences of parts.

The remainder of the paper is organized as follows: Section 2 gives a simple introduction to the printer features we consider and provides further motivation to our work. Section 3 describes the MMLS algorithm developed to address the printing direction sub-problems. These sub-problems rely on one of several objective functions described in Section 4. The approach for printing complex objects is described in Section 5. Section 6 presents and discusses the results on two case-studies. The paper is ended on Section 7 with some conclusions and remarks. An appendix is used to provide additional details on the MMLS algorithm.

2 Motivation for 3D printing of complex objects

In this section we provide additional information about the 5-axis printer setting considered in this paper. The considered printer provides the standard x, y, and z head movements and two additional degrees of freedom at the printer table, allowing for its rotation and tilt. Figure 1 shows a virtual representation of the 5–axis printer (FIBR3DEmul [9]).

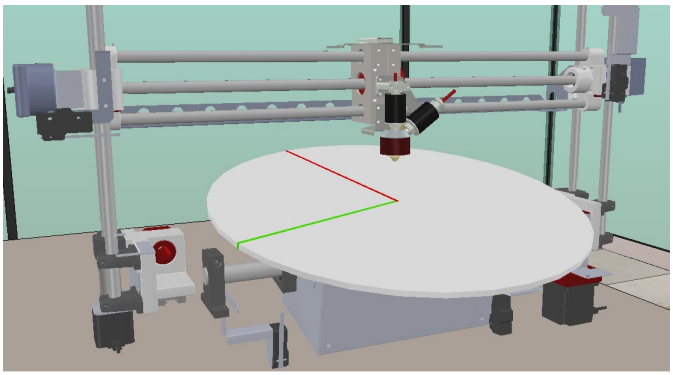


Figure 1: Virtual representation of the 5-axis printer (FIBR3DEmul [9]).

Our approach considers a decomposition of complex objects into simpler parts. Taking advantage of the 5–axis printing capability, each part may be individually printed in an optimal way, thus reducing the number of supports needed in the overall object and attaining high final object quality due the lower material surface roughness. The approach can be structured in four stages: *STL input*, where the object to be printed is provided in the STL file format, possible obtained from a CAD model; *Object parts identification*, where the object is decomposed into parts; *Optimal object building sequence*, to determine the optimal building sequence of parts leading to the desired object; and *Sequence slicing and printing*, where each part is sliced accordingly to the selected rotation, and the printer path is generated.

An STL file is expected to be provided in the *STL input* stage. The STL file is a description of the model to be printed, defined by a set of triangles (facets) and a normal direction to the facet, pointing to the object exterior. Hence, the object is defined by a mesh of triangles. In the last decades a few strategies to split an object into several parts have emerged, but they often considered the angle between two consecutive facets (e.g., angles close to 90^o) to decide about possible locations for splitting the object [8]. However, this splitting strategy is not suitable for some objects, since angles

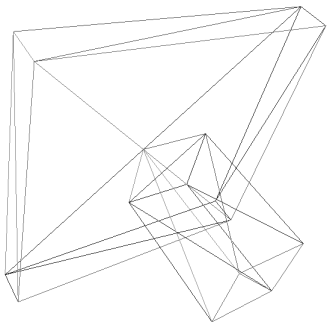


Figure 2: Object with two closed parts.

close to 90^o between surfaces in smooth objects may be defined by a high number of facets whose consecutive angles are faraway from 90^o. In the present work we assume that the object to be built is already provided as a collection of parts from the CAD model, i.e., the user builds the object CAD model by taking into consideration the object possible many parts. The object CAD model is to be exported (saved) in STL making sure that parts are closed and not merged into a single object part. For example, in SolidWorks© each part is to be designed at a 2D drawing plan and when extruding it to 3D the option *merge result* must be switched off to create a closed part (surface). See Figure 2 for an example of an object formed by two closed parts. In the *Object parts identification* stage the object is obtained by reading the provided STL file providing the closed parts of the object.

The main contribution of our paper is in the stage *Optimal object building sequence*, where we propose a strategy to compute the optimal part building sequence and corresponding orientation (rotation) of parts. The strategy takes advantage on the printer ability to rotate and tilt the printer bed and on the decomposition of the complex object into closed parts, allowing each part of the object to be printed in an optimal oriented way. The printing consists in determining when (sequence) and how (orientation/rotation) to print each part. To reduce the complexity of the sequencing optimization problem in hands, the many optimal printing directions of each part are previously addressed. The authors in [26] addressed the computation of a unique global optimum for the optimization problem of getting the optimal printing direction. However, a single optimum computation reveal itself insufficient, since the computed optimal printing direction may lead to an infeasible sequencing problem (as the unique obtained optimal printing direction may still lead to a collision when building other parts). Therefore, our optimal building planning strategy relies on the computation of all parts optimal rotations, which are later on used in the sequencing optimization problem.

After obtaining the optimal printing sequencing, the last stage (*Sequence slicing and printing*) addresses the traditional slicing and printing using the previously computed optimal sequence and parts orientation. This last stage is out of the scope of the present paper, and, therefore, not reported here in detail. However, an implementation of this stage was carried out for a 5–axis printer setting. In practice, it is only this last stage that can properly validate the proposed strategy and show that a valid path planning is indeed obtained. In this stage, given an optimal sequence and a parts' orientation, one produces the CNC (Computer Numeric Control) instructions for the printer.

In the next section, we briefly describe a strategy to compute all the optimal rotations of a given part, whose obtained solutions are to be used in the final optimal object sequencing strategy. The proposed algorithm to compute all optimal rotations is based on a multistart multisplit approach relying on a local optimization solver in order to converge to stationary points/local minimizers of the rotation optimization problem. The optimal sequencing algorithm is described in Section 5.

3 A multistart multisplit local search approach

The proposed approach to print complex objects relies on the capability of determining all possible optimal ways to print a part, w.r.t. some performance measure $f(x)$, i.e., to compute all the local optima of a part rotation optimization problem. In this section, we address a general optimization algorithm for the bound constrained minimization of a performance measure $f(x)$ given by

$$\min_{x \in \Omega} f(x), \tag{1}$$

where $\Omega \subset R^n$ is defined by $\Omega = \{(x_1, \ldots, x_n)^T \in R^n : -\infty \leq \text{lb}_i \leq x_i \leq \text{ub}_i \leq +\infty, i = 1, \ldots, n\}$. Assumptions on the smoothness of objective function $f(x)$ are postponed to Subsection A.4 (since they will depend on the local search optimization strategy to be used). We assume it is computationally expensive to evaluate the objective function.

There are a few solvers available to address problem (1) [21]. Previous proposed approaches for the computation of all the local or global optima include methods that multistart local search algorithms (e.g., MLSL [29, 30], GLODS [6], MADS [2]), but often the local search runs do not share information among them. Concurrent evaluation of the objective function is also possible for some available solvers (e.g. pVTDirect [15, 16, 17], APPSPACK [13], and HOPSPACK [27]). We instead aim at developing a solver that incorporates both features, in other words that does concurrent function evaluations and multistart of local search algorithms, sharing information of the objective function evaluations across different searches. Forthermore, the number of simultaneous local search runs is dynamic. By constructing quadratic models of the objective function, the solver detects possible valleys of convexity of the objective function, where new local search runs are initiated. This algorithm is better suited for the application in hands, since the considered objective function in the rotation problem is expected to be convex in the vicinity of local optima and we aim to explore this local convexity.

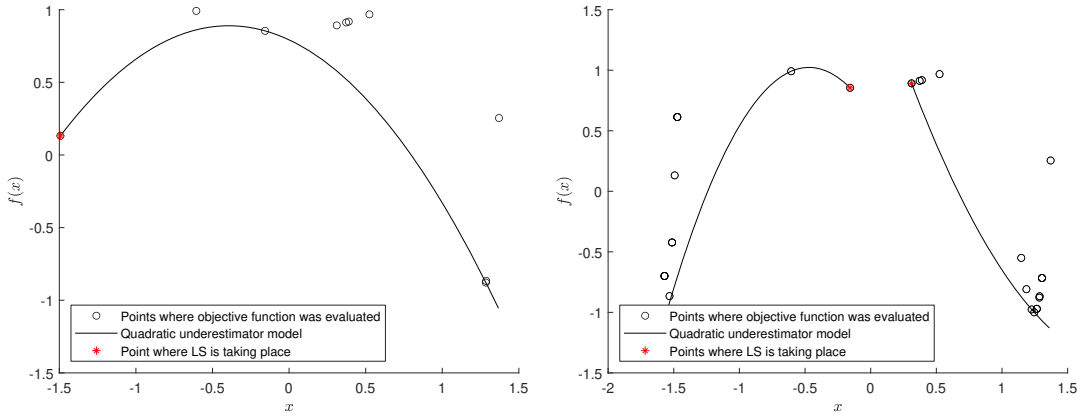
Our framework considers a certain number R of local search (LS) runs, possible made in parallel. Each LS run is guided by the run center, which consists of the point where LS is taking place. LS runs may be seen as independent runs of the local search algorithm. LS runs share information between runs, in the sense that all history of objective function evaluations is available to every LS run. Our framework is organized around inner and outer iterations. Inner iterations consist of LS iterations, while outer iterations consist of a clustering of historic objective function evaluations used to build a piecewise quadratic underestimator of the objective function. By forming clusters of points, from all the points where f has been evaluated, we are able to make a decision about the objective function landscape. The convexity information obtained from the clusters allows us to decide if a *split* of LS runs is appropriate. By splitting we mean that a new LS run will start based on the convexity information available. See Figure 3 for an outer iteration on a toy problem.

In this way, we parallelize LS runs when exploiting the objective function landscape by building underestimation models. Multistart is accomplished by starting with a set of initial points and starting new LS runs whenever the objective function landscape justifies. Unnecessary LS runs are removed/merged whenever convergence to the same local stationary points is detected.

The Multistart Multisplit Local Search (MMLS) framework is presented in Algorithm 1.

Note that each outer iteration may be done in parallel, assuming that each LS run has access to the historic data of other LS runs. Each LS run may report the objective function evaluations (evaluations possible done in parallel) and the current run center point (a point in the historical data) to a master process. In this way, Step 4 has all the ingredients to run independently for each LS run. Step 8 may also be taken in parallel as long as each LS run makes the decision to stop whenever some efficiency measure is met (e.g., either stop due to stationarity, if the run center is approaching a known stationary point, or if the run center is approaching another LS run center with sufficiently lower objective function value).

Convergence of the proposed approach is highly based on the convergence properties of the LS algorithmic choice. As long as the clustering and splitting approaches are kept finite, we note that the proposed approach generates sequence of iterates from the LS framework (one for each run center). Insuring inner iterations of the proposed approach to comply with the LS convergence rules and taking



(a) A single cluster and corresponding quadratic underestimator model ($R = 1$).

(b) Two clusters and corresponding quadratic underestimator models obtained after a split ($R = 2$).

Figure 3: MMLS outer iterations on a toy problem where $f(x) = \sin(e^{x^2})$, $-\frac{\pi}{2} \leq x \leq \frac{\pi}{2}$. LS stands for local search.

Algorithm 1 MMLS framework

1: Select initial points, x_j^0, $j = 1, \ldots, R_0$, for R_0 LS runs (and additional LS parameters).
2: **In parallel:**
3: **for** each OUTER iteration $\ell = 0, 1, \ldots$ **do**
4: Create R_ℓ clusters $C_j^\ell, j = 1, \ldots, R_\ell$, (using all the history of points available so far), taking x_j^ℓ, the run centers, as centroids, i.e., cluster R_ℓ is formed by x_j^ℓ and the historical points closer to x_j^ℓ (w.r.t. other run centers).
5: Build an underestimator quadratic model for each cluster.
6: Decide whether any LS run is split. Let $x_j^{\ell+1}$, $j = 1, \ldots, R_{\ell+1}$ be the new run centers.
7: Perform a certain number of INNER iterations for the $R_{\ell+1}$ LS runs, starting from $x_j^{\ell+1}$, $j = 1, \ldots, R_{\ell+1}$.
8: Decide whether any LS run is merged, *i.e.*, remove runs from $R_{\ell+1}$ whose run centers are too close.
9: **end for**

the new run center to be the one with the lowest objective function value when two runs are merged would most likely lead to the same convergence results of the chosen LS algorithm (and that is the case when the choice is direct search). The proposed approach details are postponed to an appendix.

4 Optimal part rotation

In this section, we use the MMLS algorithm to address the optimal part orientation/rotation. Object (or part) orientation is an important stage of the printing process, consisting of the computation of the printing direction, or, equivalently, in the computation of the object rotation while the printing direction is kept fix. There are several performance metrics that can be used to compute the optimal printing direction. We will be considering the mathematical optimization problem of determining the optimal printing direction for which the staircase effect, support area, or building time is minimized (see [26] for additional details). In the context of a standard 3D printer, where the printer is able to move along the x, y, and z axes, the printing direction corresponds to compute a vector along which slicing is to take place.

The printing direction is represented by a normalized vector described by two angles in a spherical coordinate system, i.e., one is requested to compute α (a rotation along the x axis) and β (a rotation along the y axis) angles to obtain a printing direction. For a matter of completeness, we reproduce the objective functions mathematical expressions of [26] below. The staircase effect can be measured by computing

$$\widehat{SE}(r) = \frac{t^2}{2} \sum_j \begin{cases} |d.n_j(r)|A_j, \text{ if } |d.n_j(r)| \neq 1, \\ \\ 0 \text{ otherwise,} \end{cases}$$

where t is the (constant) layers height, $d = [0, 0, 1]^\top$ is the slicing direction along the z axis, and A_j is the area of the mesh triangle j. The vector $n_j(r)$ is the normal to the mesh triangle j related to a rotation given by $r = (\alpha, \beta)$. The part area that needs to be supported can be measured by computing

$$\widehat{SA}(r) = \sum_j \begin{cases} A_j|d.n_j(r)|\delta, \text{ if } d.n_j(r) \neq -1 \text{ and } j \text{ is not at the printing table,} \\ \\ 0 \text{ otherwise,} \end{cases} \quad 4$$

where

$$\delta = \begin{cases} 1, & \text{if } d.n_j(r) < 0 \\ 0, & \text{if } d.n_j(r) > 0. \end{cases}$$

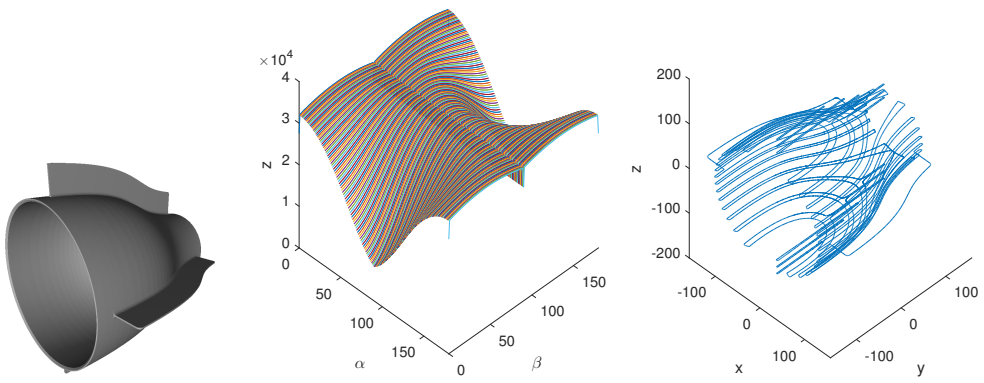
An approximation to the building time may be obtained by computing the object height along the slicing direction, leading to the following equation:

$$BT(r) = \max(d.v_1(r), d.v_2(r), ..., d.v_n(r)) - \min(d.v_1(r), d.v_2(r), ..., d.v_n(r)),$$

where $v_i(r)$, $i = 1, \ldots, n$, are the mesh triangles vertices related to a rotation given by r.

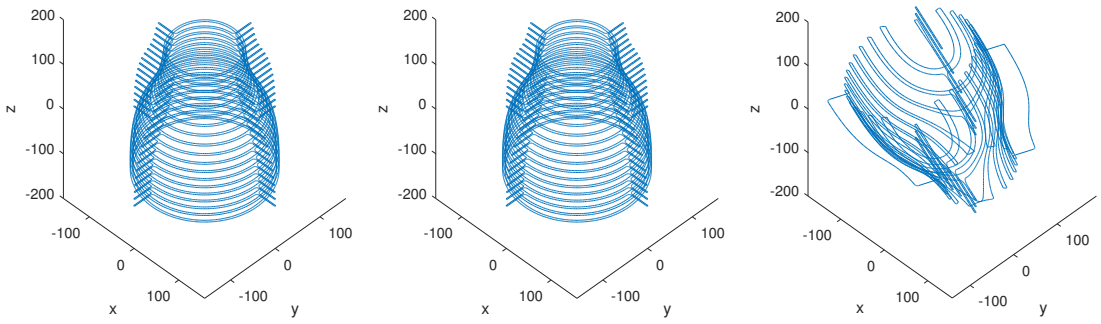
The objective function landscape of a duct object w.r.t. the $\widehat{SE}$ measure can be observed in Figure 4, as a function of α and β. Observing the function landscape, one sees that the objective function has two global minima and an infinite number of local minima at $\beta = 90^o$. This optimization problem was first solved to global optimality by considering the PSwarm solver in [26]. Since we aim to find all possible local optima, the MMLS solver was used to address the optimal printing direction of the duct object presented in Figure 4a. A maximum of 2000 functions evaluations was imposed and the solver stopped after 2115 objective function evaluations[1], after performing 80 inner iterations. Twenty points (candidates to minimizers) were obtained and four of them attained the convergence tolerance of 10^{-5} in the inner iteration, indicating that they are local optimal points. The objective function values at the obtained local optimal points allows us to conclude that two are indeed global

[1]The maximum number of function evaluations is checked at the beginning of an outer iteration (Step 3 of Algorithm 1), which may then cause the budget to be exceeded in an inner iteration.



(a) Duct object. (b) Objective function landscape. (c) Duct slices before optimization.

Figure 4: Duct object.



(a) $(\alpha, \beta) = (90^o, 0^o)$. (b) $(\alpha, \beta) = (90^o, 180^o)$. (c) $(\alpha, \beta) = (111^o, 90^o)$.

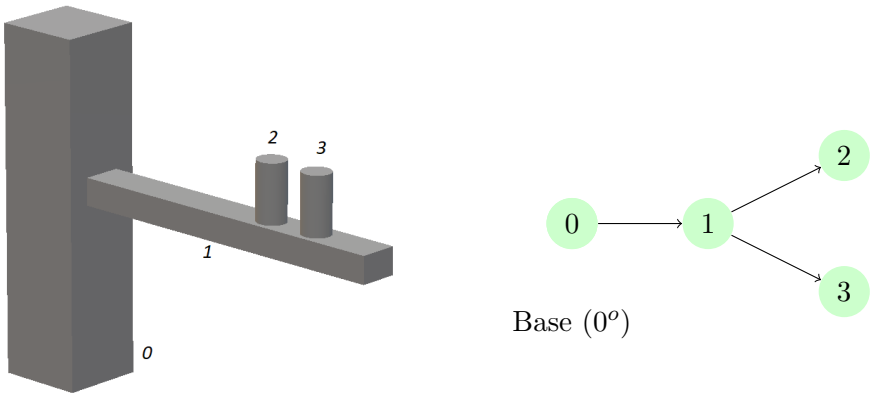
Figure 5: Duct object slices at optimal rotations.

optima and two are local optima points. We depict the two global optima and one selected local optima in Figure 5, showing that MMLS is able to compute all the global optima and some local optima for this problem.

5 Complex objects printing approach

Herein, we consider objects to be complex if they are composed by more than one closed part, where each closed part can be printed in a standard 3–axis printer individually. Any complex object can be handled by our proposed approach, provided that the object is feasible to be printed. The possibility of printing the complete object (e.g., a simple object formed by a unique closed part) in a standard 3–axis printer, possibly using supports, has no interest to the herein proposed approach.

Figure 6a illustrates a complex object available in the literature with four closed parts obtained from the STL file, from which we can establish the parts printing connections for the object and build the corresponding direct graph (Figure 6b). Without loss of generality we can assume that part 0 is connected to the printer bed (floor), while part 1 is connected with part 0, and parts 2 and 3 are connected with part 1. The object is decomposed into $T = 4$ parts. We start by introducing some notation needed to describe the sequencing optimization model and to present the proposed printing



(a) Decomposition of complex object.

(b) Graph with parts connections.

Figure 6: Complex object proposed by Ding *et al.* [8].

approach.

5.1 Variables and constraints of the model

We will assume that the set of local or global optima for the part rotation (or slicing direction) is available, obtained by using the MMLS solver described in Section 3. Let K_i be the number of known optima for the rotation of part i, $i = 0, \ldots, T-1$, i.e., let $r^*_{i,k}$, $k = 1, \ldots, K_i$, be the known optimal solutions of sub-problem (1) for part i obtained with the MMLS solver. MMLS applied to sub-problem (1) always provides at least one (approximate) local optimum, resulting in $K_i \geq 1$. We define the set of binary variables $y_{i,k}$ to be

$$y_{i,k} = \begin{cases} 1, \text{ if rotation } k \text{ of part } i \ (r^*_{i,k}) \text{ is to be considered} \\ 0, \text{ otherwise.} \end{cases}$$

Clearly we print a part once and natural constraints on the variables $y_{i,k}$ are

$$\sum_{k=1}^{K_i} y_{i,k} = 1, \qquad i = 0, \ldots, T-1. \tag{2}$$

To compute the optimal sequencing of parts we further define the $x_{i,t}$ binary variables that indicate if part $i = 0, \ldots, T-1$ is to be built at the time slot t, where $t = 0, \ldots, T-1$, i.e.,

$$x_{i,t} = \begin{cases} 1, \text{ if part } i \text{ is to be built at time slot } t \\ 0, \text{ otherwise.} \end{cases}$$

Only T time slots are necessary, since the worst case corresponds to build all parts sequentially. Clearly a part may only be built at one time slot, i.e.,

$$\sum_{t=0}^{T-1} x_{i,t} = 1, \qquad i = 0, \ldots, T-1. \tag{3}$$

To impose a precedence of parts in the building process, we need to impose two sets of constraints that must be satisfied for every part i that precedes part j:

$$\sum_{t=0}^{l} x_{i,t} \geq \sum_{t=0}^{l} x_{j,t}, \qquad l = 0, \ldots, T-1, \tag{4}$$

imposing that part i must be built at least at the same time slot, and

$$x_{i,l} \leq \sum_{k=0,k\neq l}^{T-1} x_{j,k}, \qquad l = 0, \ldots, T-1, \tag{5}$$

eliminating the possibility of building i and j at the same time slot. A strict inequality cannot be considered in (4), since it would render infeasible valid solutions. Suppose that part 1 precedes part 2 and cannot be built in time slot $t = 0$ (e.g., due to other possible parts precedences). Then the constraint $x_{1,0} > x_{2,0}$, for $l = 0$, would never be satisfied, since $x_{1,0} = 0$ and $x_{2,0} = 0$ (they cannot be built at $t = 0$), while it may still be possible to build part 1 and 2 in a later time slot.

A nonlinear black-box constraint has to be imposed to ensure that the parts building sequence (together with the parts rotation) provides a feasible building sequence, i.e., to insure that the building sequence does not provide any type of collision between the printer (head or table) and previously built parts. The constraint

$$\texttt{NoCollision}(x, y, r^*) = \text{true} \tag{6}$$

is thus imposed so that the model produces an optimal solution that leads to a building sequence that is in fact possible to be implemented, where $\texttt{NoCollision}(x, y, r^*)$ is a function returning true if x, y, and r^* do not leads to a collision. This function is printer dependent, since it uses the selected parts rotations r^* (defined by y) to decide about a possible collision of the printer head with previous built parts, when building the remaining parts (defined by x).

A second nonlinear black-box constraint

$$\texttt{NoSupport}(x, y, r^*) = \text{true} \tag{7}$$

must also be included in the model if one wishes to obtain an optimal solution that does not need the use of supports, where $\texttt{NoSupport}(x, y, r^*)$ is a function returning true if x, y, and r^* lead to a support free printing process. Similarly to `NoCollision`, this function uses the selected parts rotations r^* (defined by y) to decide about the need of a support when building a part, taking into consideration the built parts (defined by x).

Additionally, we assume that the parts on the printing table, i.e., on the base, must be built at the first time slot, and so $x_{i,0} = 1$ for all parts i attached to the printer table.

5.2 The optimization model

While the constraints in the previous sub-section provide a mathematical model for a feasible building sequence of parts, we aim to compute an optimal solution with respect to some performance measure of the printing process.

Consider $\widehat{SE}(r_{i,k})$ to be the staircase effect, $\widehat{SA}(r_{i,k})$ to be the support area, and $BT(r_{i,k})$ to be the building time of part i with rotation k (represented by $r_{i,k}$), as described in Section 4. Based on these performance measures and on the shortest building sequence to be defined below, we will formulate four objective functions to be used individually (using the one that best fits the application) or in a multi-objective formulation.

The building sequence size can be computed by

$$\text{BSS}(x) = \sum_{t=0}^{T-1} \left(t \left(\sum_{i=0}^{T-1} x_{i,t} \right) \right). \tag{8}$$

The objective function of our optimization model can take any of the forms

$$f(x, y) = \sum_{i=0}^{T-1} \sum_{k=1}^{K_i} y_{i,k} P(r_{i,k}) \quad \text{or} \quad f(x, y) = \text{BSS}(x), \tag{9}$$

where $P(r_{i,k})$ is any of the functions $\widehat{SE}(r_{i,k})$, $\widehat{SA}(r_{i,k})$, or $BT(r_{i,k})$. Our optimization model can then be stated as

$$\min_{x,y} f(x,y), \text{ subject to (2)–(7).} \tag{10}$$

While problem (10) has a linear objective function, constraints (6) and (7) are nonlinear and of a black-box type, which makes the optimization problem nonlinear with black-box type constraints over binary variables. Rigorous methods could (theoretically) be used to solve such an optimization (e.g. NOMAD [1] or even BFO [28]), but we aim for a global optima for problem (10). While evolutionary strategies for optimization could also be used, we also aim for an algorithm that efficiently computes a global optima. Therefore we propose in the next section a heuristic to address this challenging optimization problem.

5.3 A heuristic to obtain an optimal building sequence

We solve the previously described optimization problem by using a heuristic method. Our heuristic constructs all feasible points for the optimal sequencing optimization problem. The decision maker may then select an objective function among (9), or a convex linear combination of them, and based on that determine an optimal building sequence. The heuristic is presented in Algorithms 2 and 3. The input of Algorithm 2 (the main algorithm) is a list of pairs with all combinations of parts and corresponding optimal rotations, i.e.,

$$\mathcal{P}_{input} = \{(p_i, r^*_{i,k})\}, \qquad i = 0, \ldots, T-1,\ k = 1, \ldots, K_i,$$

where p_i is the part number and $r^*_{i,k}$ are the K_i determined optimal rotations for part number i. The heuristic relies on a recursive construction of lists with the main computational work done in the sub-routine **Level**, presented in Algorithm 3.

Algorithm 2 Main algorithm to enumerate all possible object printing sequences

1: Input: $\mathcal{P}_{input}$, list or pairs with parts and corresponding optimal rotations.
2: Output: $\mathcal{L}_F$, a list of lists with all the possible building sequences.
3: Initialization: Set $\mathcal{L}_F = \emptyset$, $\mathcal{L}_c = \emptyset$ (the list of parts in the current time slot is empty) and $\mathcal{L}_t = \emptyset$ (the list of parts in previous time slots is empty).
4: Main: $\mathcal{L}_F$ will be the result of the call to **Level**($\mathcal{L}_t$, $\mathcal{L}_c$, $\mathcal{P}_{input}$)

Algorithm 3 Recursive **Level** sub-routine

1: Subroutine **Level**($\mathcal{L}_t$, $\mathcal{L}_c$, $\widehat{\mathcal{P}}$)
2: **if** $\widehat{\mathcal{P}} = \emptyset$ **then**
3: // No more available parts to add to current lists
4: **if** $\mathcal{L}_c = \emptyset$ **then**
5: // The current time slot is empty, so add current sequence to the list of feasible points
6: $\mathcal{L}_F = \mathcal{L}_F \cup \{\mathcal{L}_t\}$
7: **end if**
8: return
9: **end if**
10: // Get parts not used and connected to current time slot
11: Let $\mathcal{P}_{connected}$ =**Connected**($\widehat{\mathcal{P}}, \mathcal{L}_t$).
12: **if** $\mathcal{P}_{connected} = \emptyset$ **then**
13: return
14: **end if**
15: **for** $p = (p_i, r^*_{i,k}) \in \mathcal{P}_{connected}$ **do**
16: **if** !**Collision**($\mathcal{L}_t \cup \mathcal{L}_c$, p) and !**Support**($\mathcal{L}_t \cup \mathcal{L}_c$, p) **then**
17: $\widetilde{\mathcal{P}} = \{\widetilde{p} = (p_i, r^*_{i,\bar{k}}) \in \widehat{\mathcal{P}}, \bar{k} = 1, \ldots, K_i\}$ // Prepare to exclude part remaining rotations
18: **Level**($\mathcal{L}_t$, $\mathcal{L}_c \cup \{p\}$, $\widehat{\mathcal{P}} \setminus \widetilde{\mathcal{P}}$) // Go recursively considering p in the current time slot
19: **Level**($\mathcal{L}_t \cup \{\mathcal{L}_c \cup \{p\}\}$, $\emptyset$, $\widehat{\mathcal{P}} \setminus \widetilde{\mathcal{P}}$) // Go recursively starting a new time slot
20: **end if**
21: **end for**

The **Level** sub-routine takes as input a list $\mathcal{L}_t$ of lists with the parts to be built in previous time slots (initially set as empty), a list of parts $\mathcal{L}_c$ to be built in the current time slot (initially set as empty), and $\widehat{\mathcal{P}}$ a set of parts not yet assigned to any time slot (initially set as $\mathcal{P}_{input}$). The **Level** sub-routine starts by checking the set $\widehat{\mathcal{P}}$ of not assigned parts for emptiness (line 2). If it is empty then the sub-routine ends, since no further parts are available to be assigned to the current time slot. The **Level** sub-routine is recursively called twice per sub-routine call, and in case of an empty $\widehat{\mathcal{P}}$ set the temporary list $\mathcal{L}_t$ is only saved once in $\mathcal{L}_F$ to avoid duplications, i.e., when the $\mathcal{L}_c$ list is empty. If only a feasible point were to be computed, then the full algorithm could stop right after the first assignment in line 6.

The sub-routine proceeds (line 11) by selecting parts still available in $\widehat{\mathcal{P}}$ that are physically connected to the parts to be built in previous time slots, given in $\mathcal{L}_t$. When $\mathcal{L}_t = \emptyset$, the **Connected**($\widehat{\mathcal{P}}, \mathcal{L}_t$) routine will return the parts $p \in \widehat{\mathcal{P}}$ that are physically connected to the printer table. If no parts are available, the sub-routine simply returns (meaning that such an $\mathcal{L}_t$ does not lead to a feasible point, since there exist parts to be built without a physical connection to parts already built).

For each part p in the set $\mathcal{P}_{Connected}$, the algorithm checks if a collision with parts already assigned (indicated by $\mathcal{L}_t \cup \mathcal{L}_c$) occurs and if the part p needs support to be built. The **Collision**($\mathcal{L}_t \cup \mathcal{L}_c$, p) and **Support**($\mathcal{L}_t \cup \mathcal{L}_c$, p) functions return true if there is a collision or the need of support when building part p after parts in $\mathcal{L}_t \cup \mathcal{L}_c$ were built, respectively. If no collision and no need of support is detected the algorithm proceeds recursively by considering the part p in the current time slot (line 18) and considering part p in the current time slot with an initialization of a new time slot (line 19). Since p is a pair of the part number and rotation, the set of parts available for future assignments must exclude the parts with the same part number as the one currently being assigned (available in $\widetilde{\mathcal{P}}$, see 17), i.e., a part may not be built again regardless of the rotation.

Algorithm 2 ends providing a list of lists with building sequences in $\mathcal{L}_F$. Given a list in $\mathcal{L}_F$, a correspondence to a (x^*, y^*) solution is easily obtained by setting $x^*_{i,t} = 1$ if part p_i is in the time slot t and $y^*_{i,k} = 1$ if rotation k is to be considered, with the remaining variables set to zero. A few comments about Algorithm 2 are in order.

While the algorithm constructs an enumeration of all feasible points, the number of feasible points is typically small, since the number of parts connections is also expected to be small (i.e., in the **Level** sub-routine we expect $\#\mathcal{P}_{connected}$ to be small). Additionally, the **Collision** and **Support** routines allow to identify, possibly in an early stage, infeasible points. Recall that exact or heuristic approaches to the optimization problem (10) need to check equations (6) and (7) for each x, y values, corresponding to a complete building sequence.

The procedure can be stopped prematurely if a single global optima for problem (10) is to be obtained, since a lower bound on the objective function value at a global optima is known. We have $f(x, y) \geq 0$ for all objective functions and a global optima is attained if $f(x^*, y^*) = 0$, except for the BT measure.

Additionally, we can take advantage on the printer characteristics and collision detection to further improve Algorithm 3 efficiency. We postpone this improvement to the numerical results section. The complex case-study of Section 6 helps to clarify these improvements.

5.4 Exemplifying the lists used in the heuristic

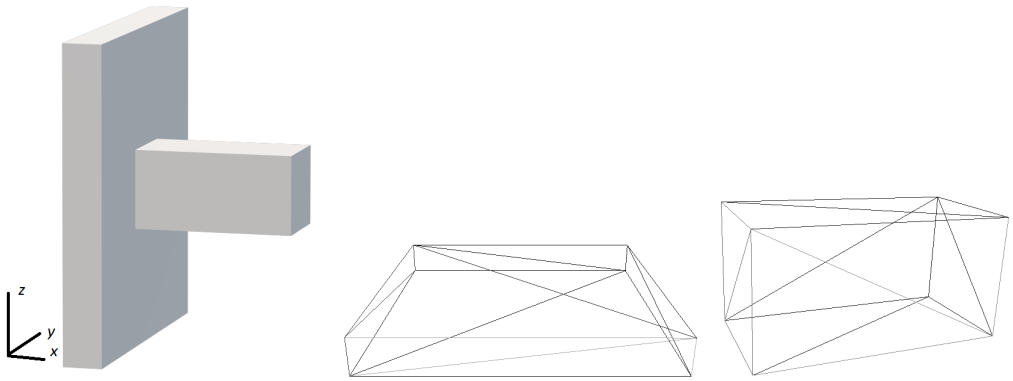
We first consider a trivial object depicted in Figure 7. Figure 7a presents the full object to be printed. Clearly this object could be printed in a standard 3D printer after a proper rotation of the object (e.g., a rotation of 0^o around the x axis and a counterclockwise rotation of 90^o around the y axis, i.e., $r = (0^o, 90^o)$). The optimal orientation of the object using the strategy proposed in [26] would lead to an optimal way to print it.

However, we are interested in illustrating that our proposed methodology also provides an optimal way to print the object. The object in Figure 7a is decomposed into two parts, presented in Figures 7b and 7c. Since each part represents a 3D rectangle, optimal part orientation w.r.t. the $\widehat{SA}$ and $\widehat{SE}$ measures leads to $\widehat{SA}(r) = 0$ and $\widehat{SE}(r) = 0$ for any rotation r corresponding to the combinations of rotating 90^o, 180^o around any axis (x or y). Considering the BT measure (measure of the part height) we have several local or global optima for each parts. Figures 7b and 7c provide a rotation that leads to global optima w.r.t. all the P measures (e.g., $r = (0^o, 90^o)$ for the first part and $r = (0^o, 90^o)$ for the second one), if parts were to be built separately.

While the object is considered trivial there are many ways to individually print the two parts. For the brevity of exposition we only consider two optimal rotations for each part, i.e., we consider $\mathcal{P} = \{(0, (0^o, 0^o)), (0, (0^o, 90^o)), (1, (0^o, 0^o)), (1, (0^o, 90^0))\}$, part 0 precedes part 1 in the building process, and part 0 is connected to the printer table.

If collisions and need of support are ignored, Algorithm 2 considers four possible scenarios given by $\mathcal{L}_1 = \{\{(0, (0^o, 0^o))\}; \{(1, (0^o, 0^o))\}\}$, $\mathcal{L}_2 = \{\{(0, (0^o, 0^o))\}; \{(1, (0^o, 90^o))\}\}$, $\mathcal{L}_3 = \{\{(0, (0^o, 90^o))\}; \{(1, (0^o, 0^o))\}\}$, and $\mathcal{L}_4 = \{\{(0, (0^o, 90^o))\}; \{(1, (0^o, 90^o))\}\}$. But $\mathcal{L}_1$ is a building sequence that leads to the need of support for part 1. Besides, a collision takes place in $\mathcal{L}_1$ since after building part 0 with $(0^o, 0^o)$, a collision occurs between the printer head and part 0, when building part 1. The same type of collision occurs in $\mathcal{L}_3$, resulting in $\mathcal{L}_F = \{\mathcal{L}_2, \mathcal{L}_4\}$. If one considers the BSS objective function in (8) then both $\mathcal{L}_2$ and $\mathcal{L}_4$ solutions attain an objective function value of 1 (two time slots) and if one considers the BT measure then $\mathcal{L}_4$ is the optimal solution because part 0 height is higher than part 0 with a $(0^o, 90^o)$ rotation (part 0 width). The $\mathcal{L}_4$ solution corresponds to the printing approach obtained by using the strategy proposed in [26].

We now consider the case-study with the complex object of Figure 6. Clearly the object cannot be printed without supports in a standard 3D printer. Again, and for the sake of simplicity, we are not considering all the local or global optima of the individual parts rotations (e.g., part 0 and part 1 have 6 optimal rotations corresponding to getting each face of the part *down*, while parts 2 and 3 have two global optima). For illustration we consider $\mathcal{P} = \{(0, (0^o, 0^o)), (0, (0^o, 90^o)), (1, (0^o, 0^o)), (1, (0^o, 90^o)), (2, (0^o, 0^o)), (3, (0^o, 0^o))\}$ and precedences given in Figure 6b.



(a) 3D composed object. (b) First part optimal rotation w.r.t. any objective function. (c) Second part optimal rotation w.r.t. any objective function.

Figure 7: First case-study with two simple parts.

Algorithm 2 would report 16 building sequences, if collisions and need for support are ignored: $\mathcal{L}_1 = \{\{(0,(0^o,0^o))\};\{(1,(0^o,0^o))\};\{(2,(0^o,0^o)),(3,(0^o,0^o))\}\}$, $\mathcal{L}_2 = \{\{(0,(0^o,0^o))\};\{(1,(0^o,0^o))\};\{(2,(0^o,0^o))\};\{(3,(0^o,0^o))\}\}$, $\mathcal{L}_3 = \{\{(0,(0^o,0^o))\};\{(1,(0^o,0^o))\};\{(3,(0^o,0^o)),(2,(0^o,0^o))\}\}$, $\mathcal{L}_4 = \{\{(0,(0^o,0^o))\};\{(1,(0^o,0^o))\};\{(3,(0^o,0^o))\};\{(2,(0^o,0^o))\}\}$, $\mathcal{L}_5 = \{\{(0,(0^o,0^o))\};\{(1,(0^o,90^o))\};\{(2,(0^o,0^o)),(3,(0^o,0^o))\}\}$, $\mathcal{L}_6 = \{\{(0,(0^o,0^o))\};\{(1,(0^o,90^o))\};\{(2,(0^o,0^o))\};\{(3,(0^o,0^o))\}\}$, $\mathcal{L}_7 = \{\{(0,(0^o,0^o))\};\{(1,(0^o,90^o))\};\{(3,(0^o,0^o)),(2,(0^o,0^o))\}\}$, $\mathcal{L}_8 = \{\{(0,(0^o,0^o))\};\{(1,(0^o,90^o))\};\{(3,(0^o,0^o))\};\{(2,(0^o,0^o))\}\}$, $\mathcal{L}_9 = \{\{(0,(0^o,90^o))\};\{(1,(0^o,0^o))\};\{(2,(0^o,0^o)),(3,(0^o,0^o))\}\}$, $\mathcal{L}_{10} = \{\{(0,(0^o,90^o))\};\{(1,(0^o,0^o))\};\{(2,(0^o,0^o))\};\{(3,(0^o,0^o))\}\}$, $\mathcal{L}_{11} = \{\{(0,(0^o,90^o))\};\{(1,(0^o,0^o))\};\{(3,(0^o,0^o)),(2,(0^o,0^o))\}\}$, $\mathcal{L}_{12} = \{\{(0,(0^o,90^o))\};\{(1,(0^o,0^o))\};\{(3,(0^o,0^o))\};\{(2,(0^o,0^o))\}\}$, $\mathcal{L}_{13} = \{\{(0,(0^o,90^o))\};\{(1,(0^o,90^o))\};\{(2,(0^o,0^o)),(3,(0^o,0^o))\}\}$, $\mathcal{L}_{14} = \{\{(0,(0^o,90^o))\};\{(1,(0^o,90^o))\};\{(2,(0^o,0^o))\};\{(3,(0^o,0^o))\}\}$, $\mathcal{L}_{15} = \{\{(0,(0^o,90^o))\};\{(1,(0^o,90^o))\};\{(3,(0^o,0^o)),(2,(0^o,0^o))\}\}$, and $\mathcal{L}_{16} = \{\{(0,(0^o,90^o))\};\{(1,(0^o,90^o))\};\{(3,(0^o,0^o))\};\{(2,(0^o,0^o))\}\}$.

Assuming that part 3 cannot be built after building part 2 (because the printer head will collide with part 2 when building part 3 due to not enough space between parts), and vice-versa, the sequences $\mathcal{L}_2$, $\mathcal{L}_4$, $\mathcal{L}_6$, $\mathcal{L}_8$, $\mathcal{L}_{10}$, $\mathcal{L}_{12}$, $\mathcal{L}_{14}$, and $\mathcal{L}_{16}$ are not feasible. The sequences $\mathcal{L}_1$, $\mathcal{L}_3$, $\mathcal{L}_9$, and $\mathcal{L}_{11}$ are also not feasible due to the need of supports when building part 1.

Hence, Algorithm 2 terminates with $\mathcal{L}_F = \{\mathcal{L}_5, \mathcal{L}_7, \mathcal{L}_{13}, \mathcal{L}_{15}\}$, where $\mathcal{L}_5 = \mathcal{L}_7$ as they correspond to the same building sequence and the same for $\mathcal{L}_{13} = \mathcal{L}_{15}$.

The sequence $\mathcal{L}_5$ corresponds to build part 0 without any rotation, apply a rotation of 90^o to build part 1, and to build simultaneously parts 2 and 3 without any rotation. Sequence $\mathcal{L}_{13}$ corresponds to rotate parts 0 and 1 and build parts 2 and 3 without rotation. Sequences $\mathcal{L}_5$ and $\mathcal{L}_{13}$ provide the same objective function values for BSS, $\widehat{SA}$, and $\widehat{SE}$ measures, being sequence $\mathcal{L}_{13}$ optimal w.r.t. the BT measure.

6 Results

In this section we illustrate Algorithm 2 applied to a more complex case-study. The object to be printed is a candelabrum with eight arms attached to a pedestal made of a circular base, depicted in Figure 8. Each arm is composed by two parts with different inclinations. Four arms are described by approximately 4000 facets (2000 facets for each part), making them to appear smooth. The other four arms are described by 24 facets (12 facets for each part) whose wire frame is clear visible in the figure. The pedestal base is a cylinder described by 740 facets and a stem described by 12 facets. The complete object is described by 15190 facets.

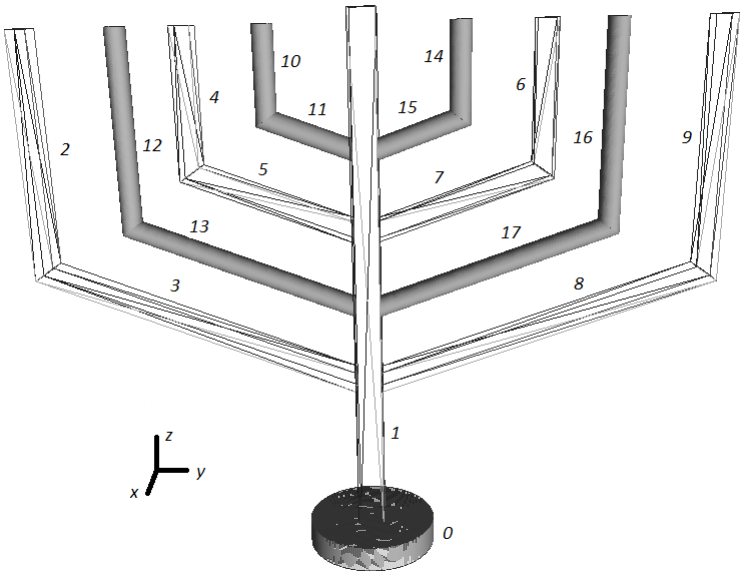


Figure 8: Candelabrum used as a complex case-study.

First, in Sub-section 6.1, we describe the application of Algorithm 1 to find as many optimal rotation of parts as possible. In Sub-section 6.2, we report how Algorithm 2 has computed the optimal sequence of parts. We finish this section with an illustration of the printing approach of the complex object in our 5–axis printer setting.

6.1 Applying MMLS to parts

In this subsection we report on the numerical results of the MMLS solver (Algorithm 1) applied to the optimization problem

$$\min_{r=(\alpha,\beta)\in[0,180]^2} \widehat{SE}(r), \tag{11}$$

where we aim to compute all the local or global optima, to be later used in Algorithm 2.

The MMLS considers $\max_{i\_iter} = 10$, $\max_{o\_iter} = 1000$, $\alpha = \max(ub - lb)/500$, and $\alpha_{tol} = 10^{-5}$ (see Algorithm 4, a more detailed version of Algorithm 1), and the nonconvex piecewise quadratic interpolation described in Section A.2. We restrict the MMLS solver to a maximum of 20 simultaneous LS runs and a maximum of 10000 objective function evaluations. An initial set of $n_i = 10$ points is considered. Optimization problem (11) is solved for each part of the object. For the sake of brevity we report the numerical results for the pedestal base (part 0), pedestal stem (part 1), non-smooth arm part (part 3), and smooth arm part (part 11). For this particular application, MMLS was given four initial guesses, namely $(0, 0)$, $(180, 180)$, $(0, 90)$, and $(90, 0)$.

6.1.1 Pedestal base (part 0)

The 3D printing procedure starts by reading a STL file with the object representation, where objects are in a certain position in space. The $\widehat{SE}$ objective function landscape depicted in Figure 9 is dependent on the part position in space defined at the design stage. In this particular case, the pedestal base is positioned faced down, i.e., the circular planar face of the pedestal base is parallel to the printing table. Clearly, if $(\alpha, \beta) = (0, 0)$, we have no staircase effect since both circular planar faces (bottom and top) are perpendicular to the z-axis, and remaining surface facets are parallel to the z-axis. Any slightly rotation along the x- or y-axes will provide a high increase in the staircase

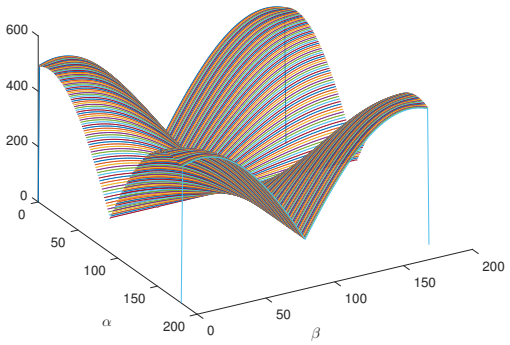


Figure 9: Pedestal base (part 0) $\widehat{SE}$ objective function.

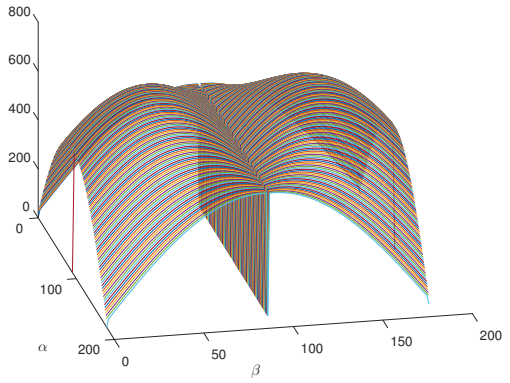


Figure 10: Pedestal stem (part 1) $\widehat{SE}$ objective function.

effect. This justifies the non-smoothness of the objective function for this particular example, which actually poses difficulties to any derivative–based solver.

With the settings previously described, MMLS reports 23 local optima (local search stopped with $\alpha < 10^{-5}$ for all reported minimizers) after 2078 objective function evaluations. By inspecting the objective function values one finds that two global optima are obtained (corresponding to $(\alpha, \beta) = (0, 0)$ and $(180, 180)$). The remaining solutions are combinations of $\alpha = 90$ or $\beta = 90$. The optimal value is approximately 125.

6.1.2 Pedestal stem (part 1)

The $\widehat{SE}$ objective function for the pedestal stem is depicted in Figure 10. MMLS stops after 6240 objective function evaluations, reporting six global optima and fourteen local optima with objective function value of approximately 500. By inspecting the objective function values of the reported points, we identified the global minima $(\alpha, \beta) = (0, 0)$, $(\alpha, \beta) = (0, 90)$, $(\alpha, \beta) = (90, 0)$, $(\alpha, \beta) = (180, 0)$, $(\alpha, \beta) = (0, 180)$, and $(\alpha, \beta) = (180, 180)$.

6.1.3 Non-smooth arm part (part 3)

The $\widehat{SE}$ objective function for the non-smooth arm part is depicted in Figure 11. This part is described by 12 facets and is attached to the pedestal stem. MMLS stops after 6665 function evaluations. It

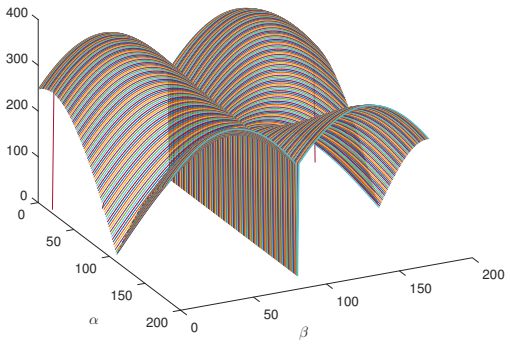


Figure 11: Non-smooth part (part 3) $\widehat{SE}$ objective function.

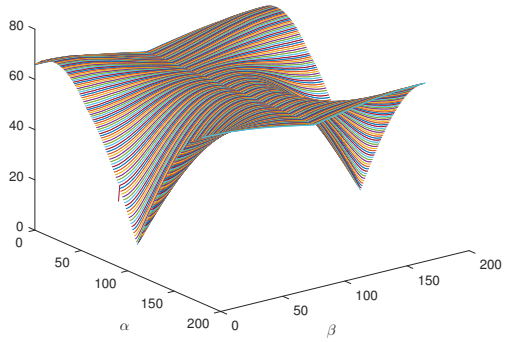


Figure 12: Smooth part (part 11) $\widehat{SE}$ objective function.

reports seventeen points attaining the requested accuracy ($\alpha < \alpha_{tol}$). By observing the objective function values we conclude that one global optima ($(\alpha, \beta) = (0, 90)$) and two local optima were found with an objective function value of approximately 15, corresponding to the points $(\alpha, \beta) = (110, 0)$ and $(\alpha, \beta) = (110, 180)$. The remaining fourteen points are local optima points with $\beta = 90$, with objective function value of 240 and two points with objective function value around 247.

6.1.4 Smooth arm part (part 11)

The $\widehat{SE}$ objective function for the smooth arm part is depicted in Figure 12. Part 11 is described by 1676 facets corresponding to a smooth arm part of the candelabrum. MMLS is able to compute the global optima with objective function value of 0 and two local optima with objective function value of 11.88 for $(\alpha, \beta) = (109.95, 180)$ and $(\alpha, \beta) = (109.95, 0)$. The remaining fifteen local optima led to an objective function value of approximately 60 corresponding to points with $\beta = 90$. These numerical results were attained after 6850 objective function evaluations.

6.2 Obtaining the optimal sequencing

Figure 13 depicts the candelabrum parts number and their connections (the part numbers are given in Figure 8). As already stated, the parts numbers correspond to the order they are found in the STL file, without any other meaning. The connections were detected by inspecting which facets are

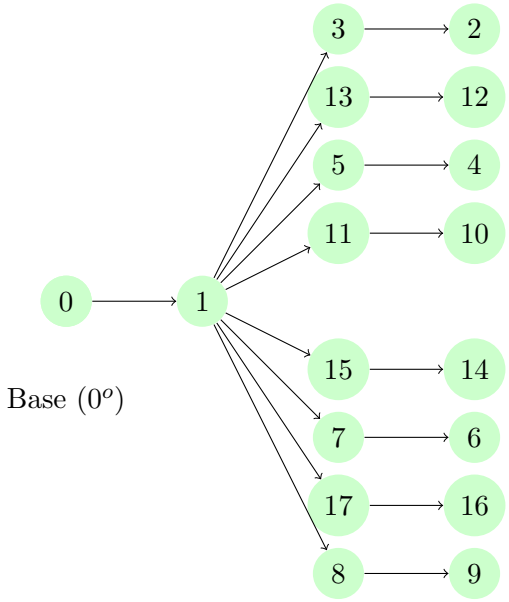


Figure 13: Graph with parts connections for the candelabrum object.

side-by-side with other parts or with the printing table.

Algorithm 2 does not incorporate a metric about the sequencing performance as it merely computes all feasible sequencing plans. As a sequencing performance metric, one can consider summing all the individual parts contribution for the staircase effect. One could also take into consideration additional penalties for building simultaneous parts (since additional movements from the printer are needed) or for positioning the printer head in a safe position before changing printing parts.

Algorithm 2 is designed to compute all feasible sequences, but it can be stopped prematurely to provide as many feasible sequences as needed. At line 6 of Algorithm 3, one can account for the $\mathcal{L}_f$ size and stop the procedure when desired. Prematurely ending Algorithm 2 can lead to a non optimal sequencing. However, Algorithm 3 can be implemented to prioritize some type of solutions. For example, the optimal rotations for each part i can be sorted by objective function value ($f(r^*_{i,k_1}) < f(r^*_{i,k_2})$, $k_1 < k_2$, $k_1, k_2 = 1, \ldots, K_i$), allowing rotations with lower objective function values to be considered first. Also, lines 18 and 19 of Algorithm 2 can be swapped if one wishes to prioritize single parts to be built at each time slot, i.e., considering the possibility of more than one part to be later built at the same time slot.

Taking advantage of the printer settings, we can go further ahead and check if the optimal rotations found for each part leads to the same printer setup (by computing the corresponding printer table rotation and tilt), discarding repeated ones. This pre-processing stage can discard some rotations found by the MMLS solver, leading to less combinations to be tested by Algorithm 2. To further improve Algorithm 2 (and make it viable for our application) one needs to take advantage of the **Collision** procedure. This procedure must be as computationally light as possible, avoiding unnecessary evaluations, and allowing the identification of infeasible sequences as early as possible. In order to make it computationally lighter, before calling Algorithm 2, all parts rotations are verified for collisions between the printer table and the printer head. Rotations that provide such a collision are not considered, since they cannot make part of a feasible sequencing. Additionally, if all optimal rotations of a given part lead to a collision with parts in $\mathcal{L}_t$, then the $\mathcal{L}_t$ sequencing can be stopped. Suppose that $\mathcal{L}_t = \{\{0\}, \{1\}, \{3\}, \{2\}\}$[2] and all possible rotations of part 5 lead to a collision, then a feasible sequencing cannot be obtained from $\mathcal{L}_t$, since part 5 must be included in the sequencing and

[2]Corresponding part rotations in the sequencing are not include for the sake of brevity.

Part number	number of o.f.e.	minimum o.f.v.	number of optima found	number of selected optima
0	2078	0	23	1
1	6240	0	20	1
2	7252	0	20	7
3	7275	0	17	4
4	4215	0	19	7
5	5403	0	18	5
6	5181	0	21	7
7	3668	0	20	5
8	3656	0	17	6
9	6469	0	20	7
10	10110	148.127	12	6
11	10316	297.078	15	7
12	10109	148.211	8	4
13	10076	297.876	15	8
14	1402	148.115	4	3
15	6616	297.009	17	4
16	1983	148.211	4	3
17	9128	297.881	18	8

Table 1: Numerical results for Algorithm 1 and 2. The *o.f.v.* is rounded to zero when lower than 10^{-5}.

a collision will be attained even if part 5 would be included later. In fact, a more sophisticate strategy can be used since the **Level** routine can return to a previous state by accounting for the parts that lead to a collision. Suppose that $\mathcal{L}_t = \{\{0\}, \{1\}, \{3\}, \{2; 5\}, \{4\}, \{7\}, \{6\}, \{8\}, \{9\}\}$ and part 11, to be included, leads to a collision with part 1, 4, and 7 (with possible different rotations w.r.t. each part). Clearly parts 6, 8, and 9 are irrelevant for the sequencing to provide a collision, i.e., a sequencing starting with $\{\{0\}, \{1\}, \{3\}, \{2; 5\}, \{4\}, \{7\}\}$ will always provide a collision with part 11. For this case the **Level** routine can return to where part 7 was selected (replacing it with another part available). Returning to any earlier part (1 or 4) is not appropriate, since the part 11 rotation that led to a collision between part 11 and part 7 may not lead to a collision with other, not considered, parts. A possible speedup strategy for Algorithm 3 is to consider a cache for collision evaluations, specially in cases where parts are defined by a huge number of facets. This would avoid the need to reevaluate collisions between parts.

6.3 Printing the object

In this section we report the numerical results obtained with the proposed framework. Table 1 presents the numerical results for the MMLS runs on each part (*Part number* in table). The number of objective function evaluations (*o.f.e.*) is reported in the second column, while the lower objective function value (*o.f.v.*) obtained is reported in the third column. MMLS found the number of optima reported in the forth column, and Algorithm 2 uses the number of optima reported in the last column (since these latter ones are those that correspond to a unique printer configuration not leading to a printer table or head collision). **Support** subroutine was not taken into consideration. One can assume that a proper rotation of parts eliminates the need for supports.

Algorithm 2, fed with the global and local minimizer in Table 1, returns as the first sequence the

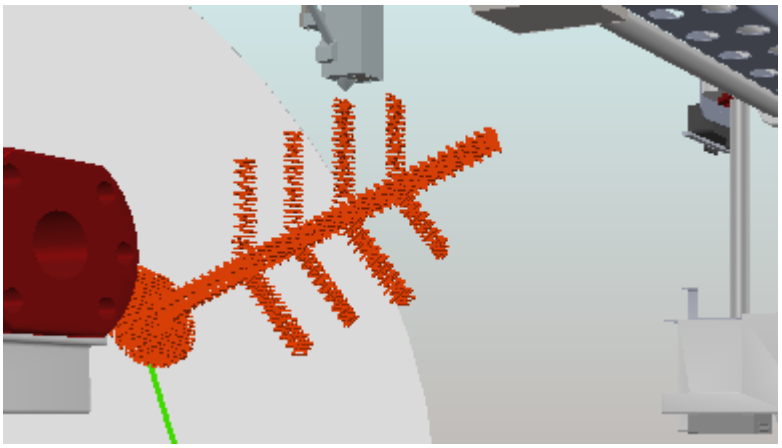


Figure 14: Simultaneous printing of parts 3, 5, 7, 8, 11, 13, 15, and 17 of the $\mathcal{L}_1$ sequencing.

following list[3]:

$$\begin{aligned}\mathcal{L}_1 = \{&\{0,(0,0)\},\{1,(0,0)\},\\ &\{3,(70,-90);5,(70,-90);7,(70,90);8,(70,90);\\ &11,(70,-90);13,(70,-90);15,(70,90);17,(70,90)\},\\ &\{2,(0,0);4,(0,0);6,(0,0);9,(0,0);10,(0,0);12,(0,0);14,(0,0);16,(0,0)\}\}.\end{aligned}$$

This sequence indicates that part 0 should be built with a rotation of $(0,0)$, followed by part 1 also with a rotation of $(0,0)$. Since Algorithm 2 prioritizes parts to be built in the same time-slot, parts 3, 5, 7, 8, 11, 13, 15, and 17 are to be built simultaneous taking into consideration the rotations that do not lead to printing collisions. Figure 14 and the remaining figures were obtained with the FIBR3DEmul [9] software package and a slice width of 2.5mm for an extrusion diameter of 0.5mm (in order to obtain an affordable simulation time). In the last time-slot, parts 2, 4, 6, 9, 10, 12, 14, and 16 are built simultaneously (see Figure 15) leading to the final Candelabrum object (Figure 17 depicts it for other sequencing). While this building sequence minimizes the staircase effect, building parts simultaneously may lead to a longer printing time, since the printer is requested to move between each part slices.

Algorithm 2 can be adapted to give priority to individual parts building (by swapping line 18 with line 19). With this algorithm change, the first sequence returned by Algorithm 2 is:

$$\begin{aligned}\mathcal{L}_2 = \{&\{0,(0,0)\},\{1,(0,0)\},\{3,(70,-90)\},\{5,(70,-90)\},\{7,(70,90)\},\{8,(70,90)\},\\ &\{11,(70,-90)\},\{4,(0,0)\},\{10,(0,0)\},\{13,(70,-90)\},\{2,(0,0)\},\{12,(0,0)\},\{15,(70,90)\},\\ &\{6,(0,0)\},\{14,(0,0)\},\{17,(70,90)\},\{9,(0,0)\},\{16,(0,0)\}\}.\end{aligned}$$

This sequence uses the same part rotations as the previous one, but single parts are firstly attempted at each time slot before considering the possibility of building multiple parts at the same slot.

If the possibility to build parts simultaneously is removed, i.e., if we remove line 18 in Algorithm 3, the same $\mathcal{L}_2$ sequence is obtained.

While these building sequences use the same rotation for each part as in $\mathcal{L}_1$, the sequences now considers the parts to be individually printed, since they belong to individual time slots. Part 3 is to be printed after part 0 and part 1 are built, followed by part 5. Figure 16 presents the printing of part 5, and Figure 17 presents the final object print.

[3]These are the reported angles (in degrees and rounded to integer) corresponding to the printer table tilt and rotation, respectively.

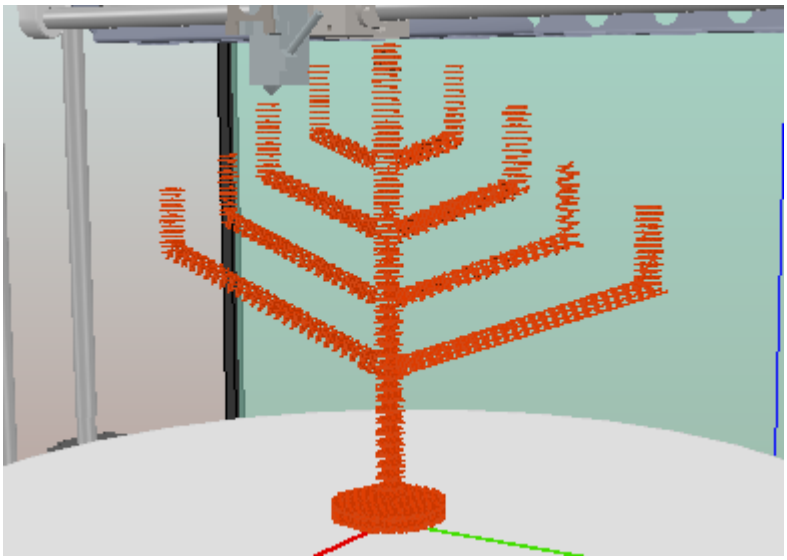


Figure 15: Simultaneous printing of parts 2, 4, 6, 9, 10, 12, 14, and 16 of the $\mathcal{L}_1$ sequencing.

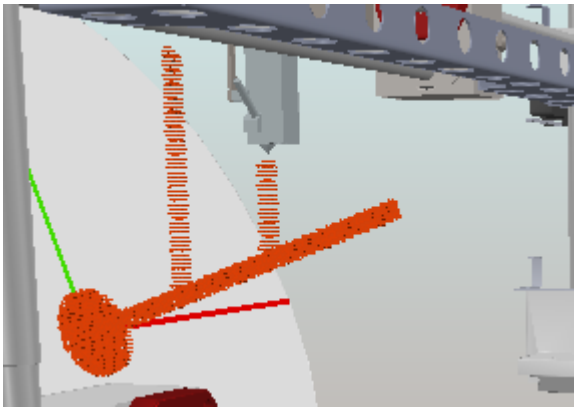


Figure 16: Printing of part 5 for the $\mathcal{L}_2$ sequencing.

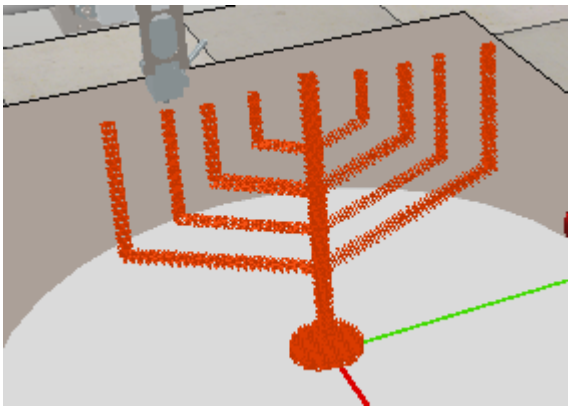


Figure 17: Final printing of the $\mathcal{L}_2$ sequencing.

The computation time to obtain a printing sequence is related to the number of facets used to describe the object parts and the number of optimal rotations of each part. The MMLS solver considers an objective function that computes the part rotation w.r.t. (α, β) during each evaluation. For a part with n facets, $3n$ points (vertices) are rotated before evaluating the objective function. The sequencing algorithm also depends on the number of facets used to describe the object, since function `NoCollision` needs to check for possible collisions between the printer head and previous built parts (each part facet is checked for collision with the printer head). While the number of optimal rotations found by MMLS affects the computational time of Algorithm 2, the major computational work is done in the `NoCollision` function. The computation time needed to obtain the herein reported sequencing was less than 2 minutes in a standard laptop computer, which is negligible when compared to the printing time.

7 Conclusions

Additive manufacturing, also known as 3D printing is an innovating technology that allows the manufacturing of complex objects at a low cost (e.g., prototyping, material, energy). By taking advantage of a 5–axis printer, we propose a strategy to print complex objects by decomposing them into simpler parts. A heuristic was proposed to address the optimal parts printing sequence, taking into consideration the optimal printing direction of each part. This strategy is shown to provide optimal printing sequences of parts and avoid support for overhang parts.

The optimal printing sequence relies on the previous solution of auxiliary sub-problems consisting of: minimizing of the staircase effect, the support area, or the building time of each part forming the object. A multistart multisplit local search (MMLS) was proposed to compute the possible many local or global optima for each sub-problem. The MMLS solver considers a set of points where local search (LS) procedures iterate. The LS set of points is dynamically updated by using quadratic models as underestimators of the objective function, which are used to guess the objective function landscape. Based on how well these quadratic models fit the true objective function, new points are added (splitting) or removed (merging) from the LS set of points. The MMLS solver computes a set of (approximate) optimal printing directions for each part of the object, which are then used for the next stage of finding an optimal printing sequence of parts.

The computation of an optimal printing sequence may be formulated as a nonlinear optimization problem with black-box type constraints over binary variables. However, solving such an optimization problem poses many difficulties, and a heuristic to compute all feasible points is proposed. The heuristic takes into consideration the parts, their connections, and all their optimal printing directions (computed by MMLS). Printer characteristics and collision detection are exploited to make the heuristic computationally efficient. The collision detection process (on which the heuristic relies on) could still be further improved by considering a cache, where results from previous collision detections are saved for later use.

The MMLS and the printing sequence heuristic are illustrated with a complex case-study, showing that the proposed approach is valid and efficient for printing complex objects in the context of a 5–axis printer. The proposed strategy addresses a specific 5–axis printer in the case studies. However, this strategy can be adapted to other printer settings and requests. The `NoCollision` and `NoSupport` functions used in equations (6) and (7) are the only printer settings specific functions, since, for example, `NoCollision` needs to take into consideration the printer head size and format, and `NoSupport` takes into consideration the extruder physical limitations and type of material to be extruded.

A Appendix

In this appendix we provide some information on our MMLS framework (Algorithm 1). A MATLAB© implementation of the MMLS solver is available by sending an email to the fourth author. We will describe the four main components, namely the clustering approach, the underestimator quadratic model, the splitting and merging procedures, and the local search procedure.

A.1 The space clustering approach

The clustering approach must terminate in a finite number of operations, so that our algorithm may generate an infinite number of LS iterates converging to a point. While any clustering technique can be used, like for example the `kmeans` clustering technique, an efficient (both in term of computational efficiency and in terms of its ability to find convexity zones in the objective function landscape) approach is requested. Our multistart approach consider several runs led by run centers, and thus we can take the run centers as natural centroids for clustering. Run centers are *converging* to stationary points and are natural candidates to be in the neighborhood of local or global optima, or stationary points. By considering run centers as centroids for the clustering, we also obtain a computational advantage, since the clustering strategy amounts then to compute the distance between all the historical data and the current run centers. At each outer iteration ℓ, our approach consists in creating R_ℓ clusters by computing the distances between every point where the objective function has been evaluated and all the R_ℓ run centers. A point x, where the objective function has been evaluated, is assigned to cluster C_j^ℓ, $j = 1, \ldots, R_\ell$, if the run center j is the closest to x. Ties are broken randomly.

A.2 A nonconvex piecewise quadratic model

We aim to approximate the objective function by using an underestimate nonconvex piecewise quadratic model. This underestimate model allows for the identification of convexity valleys, where local minimizers are located. LS runs are then able to successfully identify local stationary points.

We will use a simplified technique based on the nonconvex quadratic piecewise model suggested in [23], consisting of the minimum of several quadratics. Given a sample set $X = \{x^1, \ldots, x^{n_p}\}$, a nonconvex piecewise quadratic underestimate of f (with n_q quadratics) can be obtained by solving a convex quadratic model. Although we can rewrite this convex piecewise quadratic model into another model with a linear objective, there are still several pitfalls. First, we have to solve it for global optimality to ensure effective underestimation. Then, the number of variables is excessive, since we have n_p auxiliary variables plus $(n + 1)(n + 2)/2 \times n_q$ ones, which correspond to the number of variables for the quadratic coefficients times the number of quadratics considered. Finally, the number of constraints is also high when imposing positive definiteness on all quadratics.

There are several possibilities to provide good approximations, one being to compute just a feasible point for the model. Our approach was to take a subset of $\overline{n}_p \leq n_p$ points and considering $n_q = 1$, i.e., fitting only one quadratic, resulting in a linear programming problem, where the number of variables reduces to $\overline{n}_p + (n+1)(n+2)/2$. In the context of our MMLS algorithm we consider as many subsets of points as the number of clusters, taking $\overline{n}_p = \#C_j^\ell$ at each iteration ℓ and for each cluster j, $j = 1, \ldots, R_\ell$, a subset of points consisting of those in C_j^ℓ.

A.3 LS runs splitting and merging

At each outer iteration ℓ we compute C_j^ℓ, $j = 1, \ldots, R_\ell$, clusters of points and fit R_ℓ quadratics (one to each cluster). How well the quadratic fits the cluster points allows us to conclude about the local objective function landscape. A possible measure of fitness is the sum of squares between the objective function and quadratic model values, given by

$$\theta_j^\ell = \sum_{i=1}^{(\overline{n}_p)_j^\ell} \left(f(x_j^i) - q_j^\ell(x_j^i)\right)^2, \quad j = 1, \ldots, R_\ell,$$

where $(\overline{n}_p)_j^\ell$ is the number of points x_j^i in cluster C_j^ℓ, and q_j^ℓ is the corresponding quadratic model.

We always have $\theta_j^\ell \geq 0$, and $\theta_j^\ell = 0$ if q_j^ℓ is an interpolation model. Since θ_j^ℓ gives us a measure of how well the model locally fits the objective function, we use θ_j^ℓ to decide about a new LS run. If θ_j^ℓ is large a new LS run is attempted, whose run center is a point in cluster C_j^ℓ not already used in a LS run and sufficiently away from the current run center. If such a point does not exist, then we start a new LS run using a random generated point that is sufficiently away from the current run center.

Starting a new LS run at random points sufficiently away from the current run center provides no guarantee that the LS run converges to a different point. So, we may have two LS runs converging to the same point. A LS run is stopped if some LS stopping criteria is met or if its run center is close enough to another LS run center (running or already stopped). In order to promote convergence, the LS run whose run center has the highest objective function value is the one to be stopped.

A.4 Direct search local procedure and implementation details

We assume that derivatives of the objective function in (1) are unavailable or are difficult to obtain, and we have chosen (probabilistic) Direct Search (DS) [11] as our local search (LS) method. Probabilistic DS has shown superior performance when compared to deterministic DS, and the version with two polling directions (one randomly generated and its negative) has exhibited the best performance [12]. DS is in general a robust method that handles well noise and non smoothness in the objective function. Other derivative-free methods are described in [5].

The MMLS approach of Algorithm 1 is now described in Algorithm 4 as a full implementable algorithm. We need to detail some parameter settings. DS is controlled by a step size parameter (α in our case). While typical implementation of a DS algorithm considers a high value for the initial α, in the hope not to focus right away in a local search, our goal is the opposite, since we aim for DS to converge to a local (closer to the run center) minimizer. The α parameter is thus initialized with a small value, $\alpha = \| \text{ub} - \text{lb} \|/(50R_0)$, where R_0 is the initial number of runs, and lb and ub are the vectors of lower and upper bounds on the variables in (1). The LS α parameter is also used to control if two run centers are *close* enough and to generate random points *far away* from the current run center. Since α is expected to be a small value, instead of α we use $\bar{\alpha} \gg \alpha$ such that $\bar{\alpha} = \alpha^2$ if $\alpha > 1$, $\sqrt{\alpha}$ if $\alpha < 1$, and 2α if $\alpha = 1$.

If f is Lipschitz continuous with constant L we have $\|f(x) - f(y)\| \leq L\|x - y\| \approx \mathcal{O}(\alpha)$. If q is a good approximation of f we would also expect $\|f(x) - q(x)\| \approx \mathcal{O}(\alpha)$, and, therefore we make a decision of splitting cluster j if $\theta_j^\ell > \alpha$.

Acknowledgements

This work was developed under the FIBR3D project titled *Hybrid processes based on additive manufacturing of composites with long or short fibers reinforced thermoplastic matrix* (POCI–01–0145–FEDER–016414), supported by the Lisbon Regional Operational Programme 2020, under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF). This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020.

References

[1] M.A. Abramson, C. Audet, G. Couture, J.E. Dennis, Jr., S. Le Digabel, and C. Tribes. The NOMAD project. Software available at `https://www.gerad.ca/nomad/`.

[2] Charles Audet, J.E. Dennis Jr., and Sébastien Le Digabel. Parallel space decomposition of the mesh adaptive direct search algorithm. *SIAM J. Optim.*, 19(3):1150–1170, 2008.

[3] H. Bikas, P. Stavropoulos, and G. Chryssolouris. Additive manufacturing methods and modelling approaches: a critical review. *The International Journal of Advanced Manufacturing Technology*, 83(1):389–405, Mar 2016.

[4] Jacques Calì, Dan A. Calian, Cristina Amati, Rebecca Kleinberger, Anthony Steed, Jan Kautz, and Tim Weyrich. 3d-printing of non-assembly, articulated models. *ACM Trans. Graph.*, 31(6):130:1–130:8, November 2012.

Algorithm 4 A practical MMLS framework

Choose parameters α (initial step size for DS), α_{tol} (tolerance for DS step size), n_i (the number of initial starting points), $\max_{o\_iter}$ (maximum of outer iterations), $\max_{i\_iter}$ (maximum of inner iterations), and a budget of objective function evaluations. Choose $\bar{\alpha} > \alpha$.
Randomly generate n_i feasible points and select an initial point x_1^0 for a $R_0 = 1$ LS run.
In parallel:
for each $\ell = 0, ..., \max_{o\_iter}$ **do**
    Create R_ℓ clusters $C_j^\ell, j = 1, \ldots, R_\ell$ (using all the history of points available so far), taking x_j^ℓ as centroids.
    Build the quadratic underestimators by solving ℓ linear programming problems corresponding to each cluster.
    Compute θ_j and decide if the j LS run is to be splitted, i.e., if $\theta_j > \alpha$ then start a new LS run using y as a new run center, where $y \in C_j^\ell$ and $\|y - x_j^\ell\| \geq \bar{\alpha}$ and y was drawn from an uniform random distribution and has not been used before to start a run. If such a y does not exist, then randomly generate, from an uniform random distribution, a point in Ω such that $\|y - x_j^\ell\| \geq \bar{\alpha}$. Let $x_j^{\ell+1}$, $j = 1, \ldots, R_{\ell+1}$ be the new run centers.
    Perform a maximum of $\max_{i\_iter}$ LS iterations taking $x_0 = x_j^{\ell+1}$.
    Stop if some optimality condition is met for all $j = 1, \ldots, R_{\ell+1}$ or if the budget of objective function evaluations has been reached.
    Decide whether any LS run is merged, *i.e.*, if $\|x_j^{\ell+1} - x_k^{\ell+1}\| \leq \bar{\alpha}$, for $j, k = 1, \ldots, R_{\ell+1}$, $j \neq k$, and $f(x_j^{\ell+1}) > f(x_k^{\ell+1})$, then remove LS run j.
end for

[5] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to Derivative-Free Optimization*. MPS-SIAM Series on Optimization. SIAM, Philadelphia, 2009.

[6] A. L. Custódio and J. F. A. Madeira. Glods: Global and local optimization using direct search. *Journal of Global Optimization*, 62:1573–2916, 2015.

[7] Paramita Das, Ramya Chandran, Rutuja Samant, and Sam Anand. Optimum part build orientation in additive manufacturing for minimizing part errors and support structures. *Procedia Manufacturing*, 1:343 – 354, 2015. 43rd North American Manufacturing Research Conference, NAMRC 43, 8-12 June 2015, UNC Charlotte, North Carolina, United States.

[8] Donghong Ding, Zengxi Pan, Dominic Cuiuri, Huijun Li, Nathan Larkin, and Stephen van Duin. Automatic multi-direction slicing algorithms for wire based additive manufacturing. *Robotics and Computer-Integrated Manufacturing*, 37:139 – 150, 2016.

[9] Carlos Faria, Jaime Fonseca, and Estela Bicho. FIBR3DEmul – an open-access simulation solution for 3D printing processes of FDM machines with 3+ actuated axes. *The International Journal of Advanced Manufacturing Technology*, 106(7):3609–3623, 2020.

[10] Wei Gao, Yunbo Zhang, Devarajan Ramanujan, Karthik Ramani, Yong Chen, Christopher B. Williams, Charlie C.L. Wang, Yung C. Shin, Song Zhang, and Pablo D. Zavattieri. The status, challenges, and future of additive manufacturing in engineering. *Computer-Aided Design*, 69:65 – 89, 2015.

[11] S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang. Direct search based on probabilistic descent. *SIAM J. Optim.*, 25:1515–1541, 2015.

[12] S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang. Direct search based on probabilistic feasible descent for bound and linearly constrained problems. *Computational Optimization and Applications*, 72(3):525–559, 2019.

[13] Genetha A. Gray and Tamara G. Kolda. Algorithm 856: Appspack 4.0: Asynchronous parallel pattern search for derivative-free optimization. *ACM Trans. Math. Softw.*, 32:485–507, 2006.

[14] Nannan Guo and Ming C. Leu. Additive manufacturing: technology, applications and research needs. *Frontiers of Mechanical Engineering*, 8(3):215–243, Sep 2013.

[15] Jian He, Alex Verstak, M. Sosonkina, and L. T. Watson. Performance modeling and analysis of a massively parallel direct–part 2. *The International Journal of High Performance Computing Applications*, 23(1):29–41, 2009.

[16] Jian He, Alex Verstak, L. T. Watson, and M. Sosonkina. Performance modeling and analysis of a massively parallel direct–part 1. *The International Journal of High Performance Computing Applications*, 23(1):14–28, 2009.

[17] Jian He, Alex Verstak, Layne T. Watson, and Masha Sosonkina. Design and implementation of a massively parallel version of direct. *Computational Optimization and Applications*, 40(2):217–245, 2008.

[18] Prakhar Jaiswal, Jayankumar Patel, and Rahul Rai. Build orientation optimization for additive manufacturing of functionally graded material objects. *The International Journal of Advanced Manufacturing Technology*, 96(1):223–235, Apr 2018.

[19] Hyunwoong Ko, Seung Ki Moon, and Jihong Hwang. Design for additive manufacturing in customized products. *International Journal of Precision Engineering and Manufacturing*, 16(11):2369–2375, Oct 2015.

[20] Prashant Kulkarni, Anne Marsan, and Debasish Dutta. A review of process planning techniques in layered manufacturing. *Rapid Prototyping Journal*, 6(1):18–35, 2000.

[21] Jeffrey Larson and Stefan M. Wild. Asynchronously parallel optimization solver for finding multiple minima. *Mathematical Programming Computation*, 10(3):303–332, Sep 2018.

[22] Linjie Luo, Ilya Baran, Szymon Rusinkiewicz, and Wojciech Matusik. Chopper: Partitioning models into 3D-printable parts. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 31(6), December 2012.

[23] O. L. Mangasarian, J. B. Rosen, and M. E. Thompson. Nonconvex piecewise-quadratic underestimation for global minimization. *Journal of Global Optimization*, 34(4):475–488, Apr 2006.

[24] Brandon R. Massoni and Matthew I. Campbell. A decomposition method for efficient manufacturing of complex parts. *Computer-Aided Design and Applications*, 14(6):705–719, 2017.

[25] William Oropallo and Les A. Piegl. Ten challenges in 3d printing. *Engineering with Computers*, 32(1):135–148, Jan 2016.

[26] S. Pereira, A. I. F. Vaz, and L. N. Vicente. On the optimal object orientation in additive manufacturing. *The International Journal of Advanced Manufacturing Technology*, 98(5):1685–1694, Sep 2018.

[27] T.D. Plantenga. *HOPSPACK 3.0 User Manual*. Sandia National Laboratories, Albuquerque.

[28] Margherita Porcelli and Philippe Toint. BFO, a trainable derivative-free brute force optimizer for nonlinear bound-constrained optimization and equilibrium computations with continuous and discrete variables. *ACM Transactions on Mathematical Software*, 44:1–25, 06 2017.

[29] A.H.G. Rinnooy Kan and G.T. Timmer. Stochastic global optimization methods part i: Clustering methods. *Mathematical Programming*, 39:27–56, 1987.

[30] A.H.G. Rinnooy Kan and G.T. Timmer. Stochastic global optimization methods part ii: Multi level methods. *Mathematical Programming*, 39:57–78, 1987.

[31] Ana Maria A. C. Rocha, Ana I. Pereira, and A. Ismael F. Vaz. Build orientation optimization problem in additive manufacturing. In Osvaldo Gervasi, Beniamino Murgante, Sanjay Misra, Elena Stankova, Carmelo M. Torre, Ana Maria A.C. Rocha, David Taniar, Bernady O. Apduhan, Eufemia Tarantino, and Yeonseung Ryu, editors, *Computational Science and Its Applications – ICCSA 2018*, pages 669–682, Cham, 2018. Springer International Publishing.

[32] S. Suryakumar, K.P. Karunakaran, Alain Bernard, U. Chandrasekhar, N. Raghavender, and Deepak Sharma. Weld bead modeling and process optimization in hybrid layered manufacturing. *Computer-Aided Design*, 43(4):331 – 344, 2011.

[33] Nobuyuki Umetani and Ryan Schmidt. Cross-sectional structural analysis for 3d printing optimization. In *SIGGRAPH Asia 2013 Technical Briefs*, SA '13, pages 5:1–5:4, New York, NY, USA, 2013. ACM.

[34] Anoop Verma and Rahul Rai. Sustainability-induced dual-level optimization of additive manufacturing process. *The International Journal of Advanced Manufacturing Technology*, 88(5):1945–1959, Feb 2017.

[35] W. M. Wang, C. Zanni, and L. Kobbelt. Improved surface quality in 3d printing by optimizing the printing direction. In *Proceedings of the 37th Annual Conference of the European Association for Computer Graphics*, EG '16, pages 59–70, Goslar Germany, Germany, 2016. Eurographics Association.

[36] Eric A. Yu, Jin Yeom, Cem C. Tutum, Etienne Vouga, and Risto Miikkulainen. Evolutionary decomposition for 3d printing. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pages 1272–1279, New York, NY, USA, 2017. ACM.

[37] Yicha Zhang, Alain Bernard, Ramy Harik, and K. P. Karunakaran. Build orientation optimization for multi-part production in additive manufacturing. *Journal of Intelligent Manufacturing*, 28(6):1393–1407, Aug 2017.