Science Arts & Métiers (SAM)

is an open access repository that collects the work of Arts et Métiers Institute of Technology researchers and makes it freely available over the web where possible.

This is an author-deposited version published in: https://sam.ensam.eu
Handle ID: .http://hdl.handle.net/10985/19984

To cite this version :

Benjamin VAISSIER, Jean-Philippe PERNOT, Philippe VERON - Lattice support structure discrete optimization for additive manufacturing - In: ACM Symposium on Solid and Physical Modeling (SPM) 2018, Espagne, 2018 - ACM Symposium on Solid and Physical Modeling (SPM) 2018 - 2018

Any correspondence concerning this service should be sent to the repository Administrator : scienceouverte@ensam.eu

Arts et Métiers Institute of Technology

Lattice support structure discrete optimization for additive manufacturing

Benjamin VAISSIER, Jean-Philippe PERNOT, Philippe VERON

Abstract

The emergence and improvement of Additive Manufacturing technologies allow the fabrication of complex shapes so far inconceivable. However, to produce those intricate geometries, support structures are required. Besides wasting unnecessary material, these structures are consuming valuable production and post-processing times. This paper proposes a new framework to optimize the geometry and topology of inner and outer support structures. Starting from a uniform lattice structure filling both the inner and outer areas to be supported, the approach removes a maximum number of beams so as to minimize the volume of the support. The geometry of the initial lattice structure is optimized at the beginning considering the possibilities of the manufacturing technologies. Then, the pruning of the structure is performed through a genetic algorithm, the parameters of which have been optimized through a design of experiments. The proposed approach is validated on several test cases of various geometries, containing both inner and outer parts to be supported. The generated support structures are compared to the ones obtained by commercial software.

Keywords: support structures, lattice structures, genetic algorithm, directed Steiner tree, additive manufacturing.

1. Introduction

Additive Manufacturing (AM) has taken a huge step towards industrialization over the last few years, and the area is growing rapidly [1]. This new family of manufacturing technologies enables the production of complex shaped parts, impossible to produce with traditional manufacturing processes. Thus, it plays a key role in the emergence of the latest industrial revolution, Industry 4.0, that is encouraging the integration of intelligent production systems and advanced information technologies [2]. As opposed to subtractive manufacturing methodologies, AM consists in joining materials to make objects from 3D model data, usually layer upon layer [3]. Thanks to this approach, geometries like lattice and porous structures, organic structures generated by topological optimization, parts with intricate flow channels are becoming possible and easier to manufacture.

Despite the growing interest and the apparent ease of implementation, the production of parts in additive manufacturing requires some precautions. Actually, with most of the AM technologies, the addition of support structures is required to ensure the good production of a part. Support structures can fulfill three main functions [4]: (i) sustain overhangs, bridges and islands; (ii) stiffen the part to prevent distortions; (iii) dissipate heat from thermal accumulation areas. An overhang corresponds to a surface forming an angle inferior to 45° with the horizontal plane. A bridge is a large overhanging area, generally horizontal, sustained at its two end points. An island corresponds to a material volume that will, at a certain building layer, be completely disconnected from the rest of the part and from the building platform.

This paper focuses on the sustainment function. The stiffening and dissipation requirements are further discussed in the conclusion. Support structures can be classified into two main categories:

- *removable* support structures are usually located in reachable spaces around the part and are removed after the production, during a post-processing phase. This type of support is widely used.
- *permanent* support structures are included in the final part to support internal cavities and unreachable areas after the production. This category is avoided as much as possible.

The optimization of support structures represents a great financial stake for the industry. For removable support structures, three characteristics can be optimized: volume, production time and removability. The volume of a support structure impacts the quantity of material fused during production. It also affects the production time. However, the time spent to manufacture the object also depends on the geometry of the support. This is because the scan speed is not the same for all the areas of the support, and usually the scan speed for the outline of a geometry is lower than the one for the filling of that same geometry. Finally, the ease of removal of a support decreases the finishing time, and diminish therefore the overall cost of the part. For permanent support structures, by definition, only the volume and the production time are to be optimized.

This article proposes a new framework for the optimization of support structures, based on a discrete optimization of lattice structures. Starting from a uniform

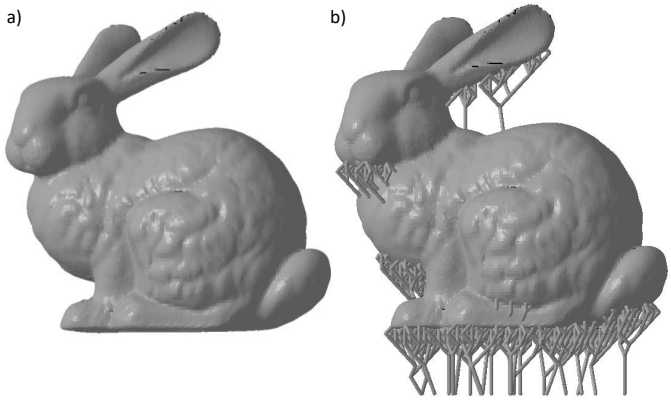

Figure 1: Optimized lattice support structures (b) generated from a triangle mesh of the Stanford Bunny (a) using the framework presented in this paper.

lattice structure filling both the inner and outer areas to be supported, the approach removes a maximum number of beams so as to minimize the volume of the support (fig. 1). The geometry of the initial lattice structure is optimized at the beginning considering the possibilities of the manufacturing technologies. Its topology is then optimized while pruning the lattice structure through a genetic algorithm. Additional post-processing steps are also performed to finalize the model and make it ready for printing.

The contribution is threefold: (i) the volume of the generated support structures is minimized, in order to reduce the overall production cost of the part; (ii) the algorithm generates aperiodic self-supporting tree-like structures, with no privilege direction, making it optimal for future mechanical optimization; (iii) the framework implements a new methodology using a genetic algorithm to solve the Directed Steiner Tree (DST) problem associated to our lattice support structure optimization.

The rest of the paper is organized as follows. After an overview of the current developments on support optimization (section 2), the section 3 describes the proposed framework composed of several steps. The problem of finding the best lattice support structure is then introduced together with the proposed genetic algorithm (GA) used for its resolution (section 4). The approach is then discussed and validated on several test cases, and the control parameters of the different steps are also optimized through several experimentations (section 5). The results are compared with the ones obtained by commercial software. Section 6 ends this paper with conclusions and perspectives.

2. Related works

Support structures are essential for the good production of parts. They prevent material collapse and part deformation. However, they represent an important proportion of the production cost of parts (e.g. material volume, production time and support removal time). Industry is thus really keen on optimizing them.

Depending on the adopted technology, the support structures do not have the same role, and therefore the same geometries. Various support structures geometries can be found in the literature. They can be classified in four main categories: extruded patterns, dually periodic patterns, triply periodic patterns and aperiodic structures.

The *extruded patterns* consist in a 2D shape in the XY plane repeated at each layer up to the part geometry. They are the most common geometries because they can be easily generated and manipulated. For example in Laser Beam Melting (LBM), Calignano carries out a design of experiments to optimize perforated blocks and lines supports with regard to part deformation, and he also proposes an orientation optimization procedure as well as a support optimization procedure [5]. Järvinen et al. optimize part surface roughness and removability of tube and web extruded structures by varying various parameters such as the thickness [6]. Jhabvala et al. generate filled block support structures with porous microstructure, thanks to a pulsed laser, making them easily removable [7]. Krol et al. developed a discrete optimization based on a Finite Element Analysis (FEA) model by subdividing extruded crossing walls [8][9]. For the Fused Deposition Modeling (FDM) technology, Jin et al. propose a slice-based algorithm to generate plastic support structures [10]. Crump et al. also filed a patent on the creation of an interface between the part and the support structures to facilitate the removal of the latter [11]. For the StereoLithography Apparatus (SLA) technology, Quian et al. developed an algorithm projecting overhanging areas onto the building platform to generate block-like support structure [12]. Finally, for the Electron Beam Melting (EBM) technology, Cheng et al. and Cooper et al. use contact-free blocks placed underneath the overhanging areas to dissipate the thermal energy induced by the process [13][14][15].

The *dually periodic patterns* consist in 3D complex shapes repeated according to a 2D pattern in the XY plane. For example in LBM, Gan and Wong optimize tree-like geometries by varying the repetition frequency in the XY plane and analyze for each support structure thus generated its influence on the temperature distribution during production, and on the surface roughness after support removal [16]. In FDM, Boyard repeats a tree-like structure under the overhanging areas to support plastic parts [17].

The *triply periodic patterns* consist in 3D complex shapes repeated in the X, Y and Z directions. For example, Hussein et al. make use of minimal surface structures (like the Schwartz diamond or the Schoen gyroid) to support a cantilever part [4][18], whereas Cloots et al. stack parts on top of each other in the building chamber by using lattice support structures [19]. In FDM, Li et al. vary the diameter of lattice structure beams in order to cre-

ate stiffer support structures [20], whereas Lee et al. propose a voxel-based hollowing method to create inner support structures [21]. In SLA, Swaelens et al. filed a patent on the geometries of support structures, including perforated crossing walls and lattice support structures [22].

The *aperiodic* support structure category gathers all the support geometries that do not present any repetition pattern. For example, in FDM, Vanek et al. compute the intersection of cones placed under overhanging surfaces to create tree-like support structures [23], whereas Zhang et al. generate lattice structures leaning on the medial axis of the part to sustain hollow part [24]. Vaidya et al. also repeat octahedral unit cells in order to create tree-like support structures sustaining overhanging areas [25].

Most of the previously mentioned articles are proposing algorithms or frameworks to support all the overhanging surfaces but fewer are taking into account the importance of minimizing the volume of the generated structures. Besides removing unnecessary beams and thus unnecessary fused material, the framework detailed in this paper operates on a pre-optimized lattice structure which geometry has the minimal volume to make it manufacturable. This results in a highly low volume support structure.

Furthermore, many approaches are exploring extruded, dually periodic and triply periodic structures but fewer are focusing on aperiodic supports. However, the overhanging and thermo-mechanical constraints of a part supporting problem do not, most of the time, follow any pattern. Therefore, the geometry of the support structures should not a priori present any shape repetition. The framework presented in this paper optimize a lattice structure by removing the unnecessary beams and therefore generates an aperiodic tree-like structure, presenting no repetition bias.

3. Overall framework

This section introduces the new optimization framework developed to sustain the overhanging surfaces of a part.

Basically, an overhanging surface is sustained if every point $\mathbf{p}_0$ of the surface is at a distance smaller than an overhang distance o_p from at least one other point $\mathbf{p}_1$ with support material directly below it. The overhang distance o_p (with p referring to process) depends on the adopted AM technology, material and print parameters. For example, with the LBM technology, it is considered that $o_p \simeq 0.5$mm. This is illustrated on figure 3 wherein green areas are sustained since they gather together points which are close enough to existing support structures. At the opposite, red zones correspond to unsustained areas far from any existing support structure. The sustainment condition can therefore be expressed as:

$$\forall \mathcal{S} \in OS(\mathcal{P}), \forall \mathbf{p}_0 \in \mathcal{S}, \exists \mathbf{p}_1 \in SP(\mathcal{S}) : \|\mathbf{p}_0 - \mathbf{p}_1\| \leqslant o_p \quad (1)$$

where $\mathcal{S}$ corresponds to an overhanging surface, $OS(\mathcal{P})$ is the set of all the overhanging surfaces of a part $\mathcal{P}$, and $SP(\mathcal{S})$ is the set of all the sustained points of $\mathcal{S}$ with support material directly below them.

From this definition, it becomes straightforward that the use of lattice structures is a good mean to ensure the sustainment condition while minimizing the support material to be used under the overhanging surfaces. The principle of the proposed framework is to generate such an initial lattice structure under the overhanging surfaces of a part and to remove from this lattice the maximum number of beams, without breaking the sustainment condition. More precisely, the proposed framework is composed of several steps illustrated on figure 2:

1. *Initial lattice generation*: starting from a watertight triangle mesh composed of one or more oriented shells, a lattice structure with a manufacturable unit cell geometry is generated to support the inner and outer overhanging areas. This lattice is obtained by repeating a parallelepipedic unit cell in the three orthogonal directions of the 3D space, with 3 constant repetition distances. In this paper, one specific unit cell geometry is used, but any other self-supporting geometry could have been chosen. The adopted unit cell combines a body-centered cubic (BCC) cell and 5 vertical beams, located at each vertical edge and at the vertical axis of the cube (fig. 4). It is defined by three parameters: a corresponds to the size of the base, h to the height of the cell, and d to the diameter of the beams. As a consequence, the maximum beam angle $\alpha_{\max}$ and the overhanging distance of the lattice o_ℓ (with ℓ referring to lattice) can be easily computed. Thus, the sustainment condition for this specific unit cell is $o_\ell \leqslant 2o_p$ which gives:

$$a \leqslant (d + 2o_p)\sqrt{2} \quad (2)$$

   Then the lattice is trimmed to the part surface, which means every beam going through the part surface is cut short. This lattice must satisfy the sustainment condition (1) because the optimization algorithm will identify the best sub-lattice of this initial lattice.

   Furthermore, the parameters of the lattice structure are not considered as variables of our lattice support structure discrete optimization. Thus, they are optimized in a preliminary step that is discussed in section 5.2.
2. *Pre-processing*: the variables and parameters of the optimization problem are identified. The nodes of the lattice connected to only one beam, also called "isolated nodes" with a valency of 1, are identified. Those nodes appear because of the trimming of the lattice to the part surface. Among the isolated nodes, the ones connected to an overhanging area are called sources and are the ones that must be sus-

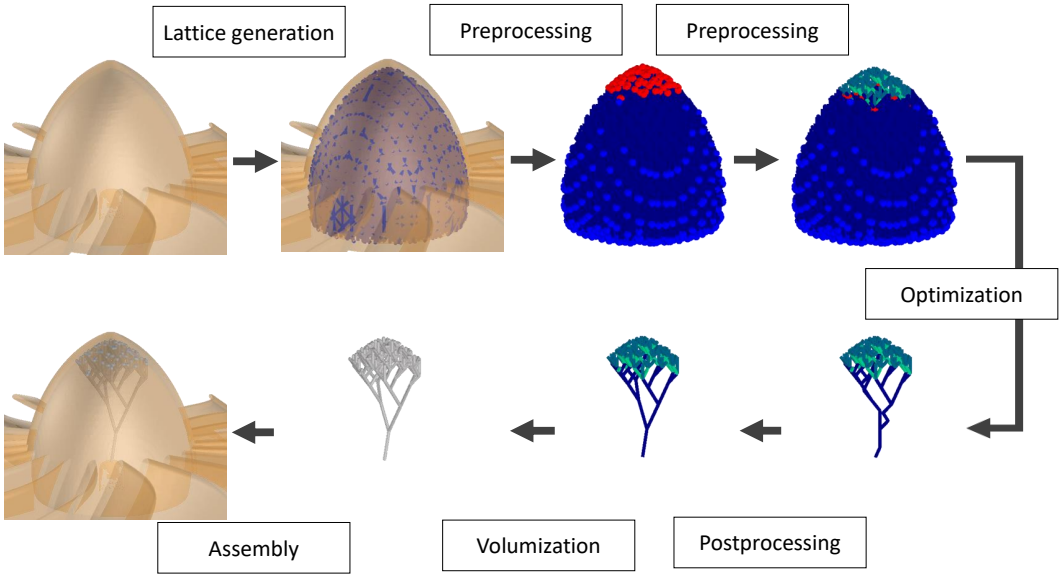

Figure 2: The proposed optimization framework (example on the plane turbine internal supports)

tained. The other isolated nodes are called wells and do not need to be sustained.

3. *Pre-optimization*: the variables of the optimization problem for which the value in the optimal solution can be deduced are isolated and removed from the set of optimization variables, thus reducing the computation time. For example, if a source (i.e. a node that needs to be sustained) has only one outgoing beam, this beam is needed to preserve the sustainment constraint and is thus necessarily part of the optimal solution. This beam is therefore directly added to the solution beam set, the source is removed from the set of variables, and the beam's lower extremity is added to the set of variables as a new source. Likewise, if one of the smallest outgoing beams is directly connected to a well (i.e. its lower extremity is a well), this beam is added to the solution beam set and the source is removed from the set of variables.
4. *Optimization*: a Genetic Algorithm (GA) is used to determine which sub-lattice of the initial lattice structure has the smallest cumulated beam length while respecting the sustainment constraint. In this step, the initial lattice structure is pruned until the stop criteria are reached. This algorithm is introduced in section 4 and the method used to optimize its parameters is discussed in section 5.1.
5. *Post-processing*: once the optimized sub-lattice is found, a post-processing step is applied to even better reduce its length. The beam paths between connection points (i.e. points of valency greater than 1) are straightened by replacing each beam path with a unique rectilinear beam between the two associated connection points. Because of the initial lattice topology, this added beam is assured to be self-sustained,

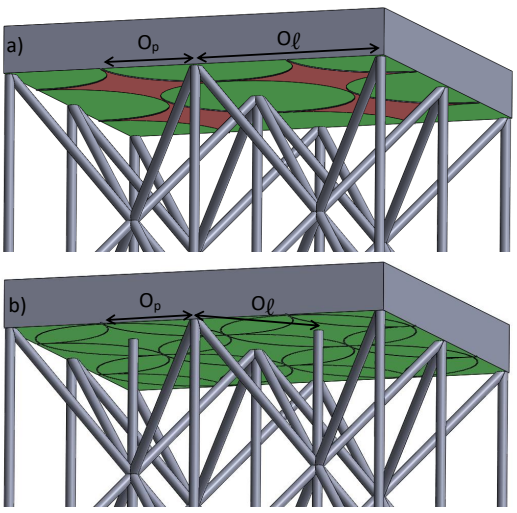

Figure 3: The sustainment condition: a partially (a) and a fully (b) sustained surface. The unsustained areas are in red.

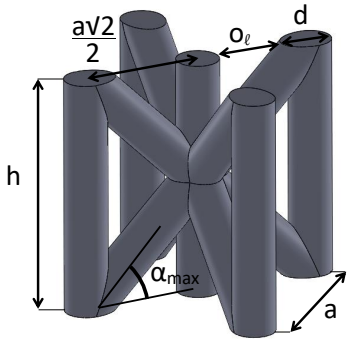

Figure 4: The unit cell used in this paper

and will not generate any additional overhanging area.

At the end of this optimization process, some preparation steps are still required before the printing. The resulting lattice structure is to be triangulated using for instance the approach of Chougrani et al. [26] to minimize the number of generated triangles. Then, the generated geometry can be sliced and the printing parameters selected.

4. Solving the Lattice Support Structure Discrete Optimization (LS^{2}DO) problem

The lattice support structure optimization problem introduced in the previous section is a discrete problem because the number of potential solutions is finite. To solve it, a Directed Acyclic Graph (DAG) is associated with the initial lattice structure and a Genetic Algorithm (GA) is used to find the subgraph respecting the sustainment constraints and with the smallest cumulated beam length.

4.1. Problem formalization

Inspired by the graph theory, a lattice $L = (N, B)$ is a set of beams B connecting a set of nodes N. The number of beams is denoted $b = card(B) = |B|$ and the number of nodes $n = card(N) = |N|$. Let us define $\mathcal{L}_3$ the set of all the lattices embedded in $\mathbb{R}^3$. For a lattice L, the preprocessing step of the proposed framework identifies the set of all the source nodes N_S and the set of all the well nodes N_W. A lattice $L' = (N', B')$ is defined as a sublattice of L if $B' \subset B$. This also implies $N' \subset N$. Let us denote $sub(L)$ the set of all the sublattices of L.

The lattice graph associated to a lattice $L = (N, B)$ is the graph $G = (V, E)$ in which each vertex in V corresponds to a node in N, and each edge in E corresponds to a beam in B. Therefore, $card(V) = |V| = n$ and $card(E) = |E| = b$. An element of V is denoted v_i with $i \in \{1, ..., |V|\}$. Similarly, a source vertex v_{si} is associated to a lattice source node, and a well vertex v_{wi} is associated to a lattice well node. Likewise, V_S represents the set of all the source vertices, and V_W the set of all the well vertices. For a vertex v_i of V, $out(v_i)$ denotes the set of outgoing edges of v_i, and $in(v_i)$ denotes the set of incoming edges of v_i. Let us denote $\mathcal{G}_3$ the set of lattice graphs generated from $\mathcal{L}_3$ and $sub(G)$ the set of all the subgraphs of G.

In the proposed approach, a lattice graph is oriented as follows. Each edge of the graph is oriented from its vertex corresponding to the highest lattice node (in the Z direction, perpendicular to the build platform) to the vertex corresponding to the lowest lattice node. Furthermore, the lattice graph is weighted with a function $w_e : E \rightarrow \mathbb{R}$ which associates to each edge of the lattice graph a cost value equal to the length of the corresponding lattice beam. For an edge e_j with $j \in \{1, ..., |E|\}$, the cost value $w_e(e_j)$ can be referred to as the length of the corresponding beam in the lattice structure. Consequently, the weight (or length) $w(G)$ of a lattice graph $G = (V, E)$ is given by:

$$w(G) = \sum_{j=1}^{|E|} w_e(e_j) \tag{3}$$

This is the objective function to be minimized when solving the LS^{2}DO problem. Indeed, the framework starts by generating a lattice structure L under the overhanging areas (red parts on the 2D Stanford Bunny of figure 5) and optimizes it by finding a sub-lattice in $sub(L)$ with the minimum cumulated beam length, that sustains all the identified sources in N_S. In terms of graph, the problem is to find a sub-graph of the initial lattice graph $G = (V, E)$ with the minimum cost $w(G)$, and which connects every source vertex in V_S to at least one well vertex in V_W. Figure 5 shows the weighted DAG associated to our LS^{2}DO problem. Red vertices are the sources, and blue vertices are the wells. The weights on the edges are equal to the lengths of the beams in the lattice structure, except the blue edges which all have a null weight and which are not associated to any beam of the lattice structure.

Now, if one connects all the well vertices to a new root vertex r, by simply adding a set of edges E' weighted with a 0 value (fig. 5), the problem remains unchanged, but it can be formulated as: "Find the tree $T = (V_T, E_T)$ in $G_{ini} = (V \cup \{r\}, E \cup E', w_e)$ with the minimal cost $w(T)$, rooted at r such that $V_S \subset V_T \subseteq V$ and $E_T \subseteq E$".

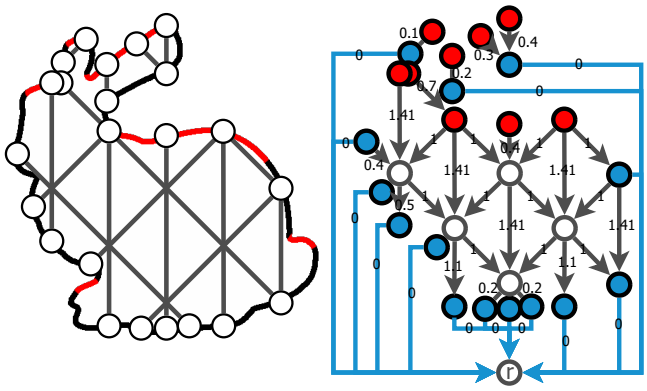

Figure 5: Example of LS^{2}DO problem on a 2D Stanford Bunny (left) and the weighted directed acyclic graph (DAG) associated (right).

This corresponds to the definition of the Directed Steiner Tree (DST) problem [27]. The DST problem is known to be NP-hard [28], i.e. no polynomial algorithm has been found to compute the optimal solution, and presumably none will ever be. Furthermore, Halperin et al. proved that, for any $\epsilon > 0$, there is no polynomial approximation algorithm within a $log^{2-\epsilon} n$ ratio, unless $P = NP$ [29].

Therefore, many subjects of research focus on approximating the solution of the DST problem. For example,

Charikar et al. presents an algorithm with an approximation ratio of $O(k^{2/3}log^{1/3}k)$, where k is the number of pairs of vertices that are to be connected [30]. Zelikovsky also proposes an $O(k^{\epsilon})$-approximation algorithm for any $\epsilon > 0$ in the case of a DAG [31].

4.2. Resolution using a Genetic Algorithm

To find an approximated solution of the DST problem associated with the lattice support structure optimization problem, a Genetic Algorithm (GA) has been set up and its control parameters have been tuned. Genetic algorithms are metaheuristics. They are generic, adaptable to various kinds of problems, and are random by nature. This randomness makes the output of every run unpredictable and potentially different from the result of a previous run. In this sense they are non-deterministic algorithms. They are part of the larger family of evolutionary algorithms.

A GA manipulates populations of chromosomes. Each chromosome encodes a potential solution to a problem, i.e. a sub-graph in the present case. To solve the LS^{2}DO problem, two encoding approaches have been considered:

- the *activation encoding* (fig. 6 left) is one of the simplest ways to encode a sub-graph into a chromosome. Here, a boolean variable x_i is associated to each edge e_i in the set of edges E of the initial graph, and these variables are concatenated to form the *activation chromosome*. If $x_i = 0$, then the edge e_i is not activated, else if $x_i = 1$ it is activated. Every sub-graph of a graph can thus be encoded.

- the *switch node encoding* (fig. 6 right) is another way to encode a solution of the DST problem into a chromosome. Here, a variable x_i is associated to each vertex v_i in the set of vertices V of the initial graph, selecting the only outgoing edge of v_i that will be activated if at least one incoming edge of v_i is activated. In the 2D example of figure 6 (right), three values can be assigned to the variables. If $x_i = 1$, then the oriented edge starting from v_i and pointing to the left is activated, else if $x_i = 2$ then the one pointing vertically downward is activated, else if $x_i = 3$ the one pointing to the right is activated. This encoding method can only encode sub-trees of a graph because for each vertex, only one outgoing edge can be activated. The set of potential solution is, therefore, the set of all the sub-trees of the initial graph rooted in r. Actually, since the solution of the DST problem must be a tree rooted in r, the *switch node encoding* is particularly adapted to solve the LS^{2}DO problem. The set of all sub-trees rooted in r is consistently smaller than the set of all sub-graphs, so the *switch node encoding* reduces greatly the computation time of the GA.

For all those reasons, in this paper, the switch node encoding has been chosen. Considering a 3D lattice structure initially composed of basic cells as the one of figure 4, the variable x_i associated to a vertex v_i of the initial graph can take five different values corresponding to the five directions of the oriented edges starting from v_i and going down. Of course, if another type of unit cell is chosen, the values of the variables are to be changed.

The execution of our GA is described in the flowchart of figure 7. At first, an initial population of chromosomes (corresponding to potential solutions) is selected. In our case, the initial population is chosen randomly but it can be selected through a heuristic algorithm, in order to obtain good initial solutions. Then the so-called *fitness* of each initial chromosome is computed in parallel. The fitness of a chromosome corresponds to its evaluation by the objective function. Here it is directly equal to the sum of the weights associated the edges activated by that chromosome. Once each chromosome is evaluated, a set of parent chromosomes is selected in the initial population. Those parents are going to make birth to the next generation of chromosomes, through crossovers and mutations. Crossover is the process of subdividing and mixing portions of at least two parent chromosomes to create at least one child chromosome, whereas mutation is the process of randomly changing gene values of a parent chromosome. However, all the selected parents are not giving birth to children: each parent has a probability to be part of a crossover, and a probability to mutate. Those probabilities are parameters of our GA. Once the child chromosomes are created, their fitness is once again computed in parallel. Some children are then selected to be reinserted into the previous population to form the next generation of chromosomes. Finally, our GA checks if the termination conditions are fulfilled by this new generation: if so, the best chromosome of the last generation is considered as the solution of the problem, and otherwise, our evolutionary algorithm goes on with a new generation created through the GA steps.

Finally, our GA makes use of 4 operators: selection, crossover, mutation and reinsertion. Moreover, our GA has 4 execution parameters that influence the algorithm: minimal size of a population (MinSize), maximal size of a population (MaxSize), crossover probability (CP) and mutation probability (MP). Thus, our GA is controlled by 8 parameters which can be optimized to solve efficiently the LS^{2}DO problem.

5. Experimentations and results

This section adresses the way the parameters of the lattice structure and the parameters of the GA can be optimized. Once the best parameters identified, the proposed approach is applied to several test cases and the generated support structure are compared to the ones obtained with commercial softwares.

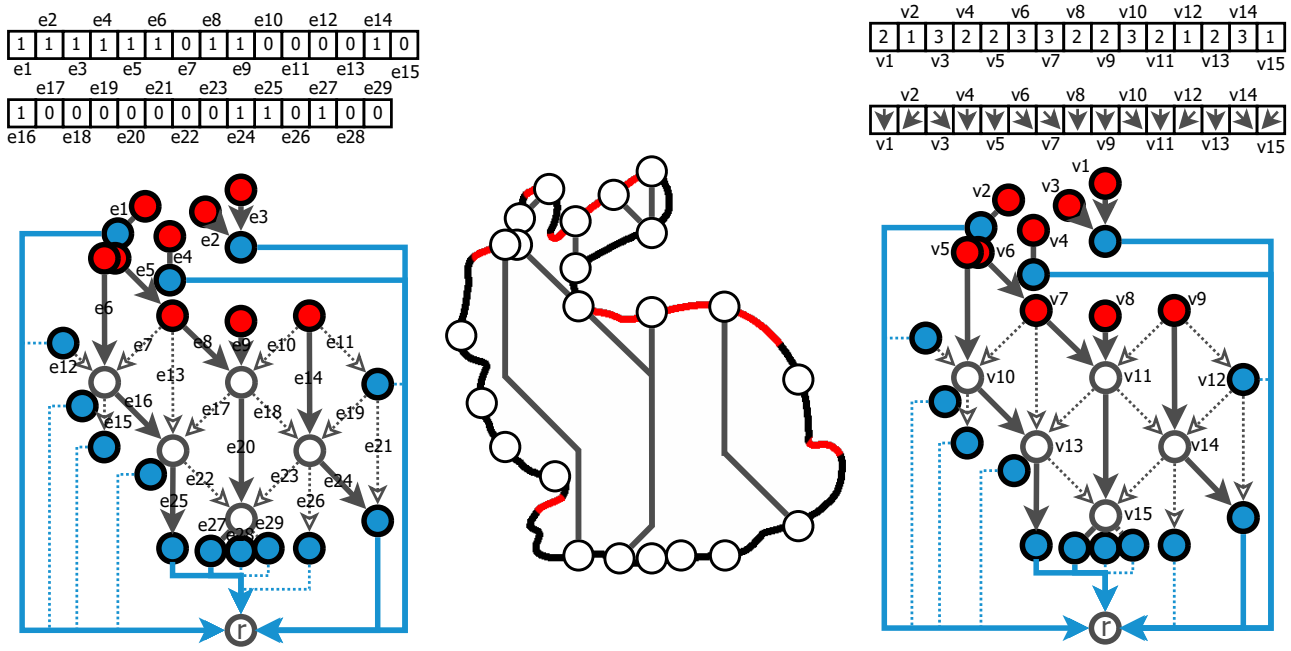

Figure 6: The activation encoding (left), the switch node encoding (right) and the resulting 2D supports (center).

Levels	0	1	2	3
Selection	Elite	Roulette Wheel	Stochastic Universal Sampling	Tournament
Crossover	One-Point	Two-Points	Uniform	Three parents
Mutation	Reverse Sequence	Twors	Uniform	–
Reinsertion	Elistist	Uniform	–	–
MinSize	50	100	500	1000
Span (MaxSize-MinSize)	50	100	500	1000
Crossover Probability	0.25	0.5	0.75	1
Mutation Probability	0.01	0.1	0.3	0.5

Table 1: Factors and their levels for the design of experiments parametrization

5.1. GA parameters optimization

The GA adopted in the proposed framework has been implemented through the GeneticSharp library, created by Giacomelli, and available on the GitHub platform. Therefore, the levels for the 4 operators of the GA steps are the ones available in this library and applicable to the DST problem. The names of these operators correspond to the ones used in the GeneticSharp library. The strategy behind each operator is not described in this paper, but it can be found with great details on the GitHub webpage. Table 1 lists the levels of the 8 parameters (4 operators and 4 each execution parameters) controlling our GA: 6 parameters with 4 levels, 1 parameter with 3 levels and 1 parameter with 2 levels have been identified as potential parameters for the DST problem.

5.1.1. Design of Experiments set up

A design of experiments (DoE) has been carried out to determine which GA parameters are the most suited for the LS^{2}DO problem. Among all the available DoE methods (e.g. full factorial, Doehlert, Box-Behnken), the Taguchi tables method has been selected because it offers a good trade-off between accuracy of the results and number of different experiments to realize. According to the number of parameters and the number of levels for each parameter, the $L_{32}(2^1 \times 4^9)$ table has been selected. It must be noted that this particular Taguchi table does not accept any 3-level parameter, whereas, for the mutation operator, only 3 levels have been identified. Therefore, the Uniform level of the mutation operator has been repeated twice. However, for the result analysis of the DoE, the effects of the 2 Uniform levels have been considered independently so that it does not affect the results. During the 32 experimentations, two quantities have been observed to define the quality of the GA parameterization: the length of the lattice structure associated to the solution graph (which need to be minimized for the LS^{2}DO problem) and the optimization time (i.e. the time before the algorithm returns a solution).

Because of the random nature of the GA, the DoE can

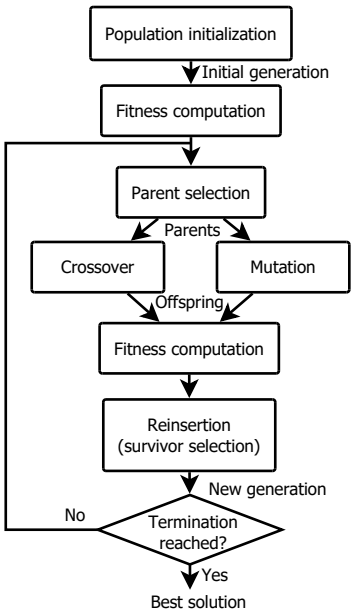

Figure 7: Genetic algorithm flowchart

be biased: one run of the GA with a certain set of parameters can produce exceptional results compared to what it would produce most of the time. In order to smooth this effect, for each set of parameters, it has been decided to run the GA 5 times, to discard the two extrema (the lowest and the highest measurements) for each measured quantity (length of the solution and optimization time), and to analyze the effects of the set of parameters with the mean of the 3 remaining measurements, called the trimmed response.

Finally, to set up the DoE, it is important to stress that the LS^{2}DO is also a part specific problem: from one part to another, the initial lattice structure is different, so the evolution of the algorithm varies. Therefore, it is not straightforward that the optimal GA parameters for one supported part are the same for another. To check this assertion, the DoE has been carried out on 3 different parts: the academic Standford Bunny, an industrial Stem and an industrial Turbina (figures 12, 13 and 14). Each part contains several more or less complex inner and outer areas to be sustained.

5.1.2. Analysis of the DoE results

Following this DoE, the 32 experimentations have been run 5 times on the 3 parts. Figure 8 compares the effects of the 3 DoE on the length measurement whereas figure 9 compares the effects of the 3 DoE on the time measurement. As a reminder, the length measurement represents the weight of the graph (equal to the length of the corresponding lattice) at the end of a run of the GA with a specific bundle of parameters. Therefore, the objective of the DoE is to find out the parameters which optimize both the length and the optimization time. For each parameter, the optimal level is thus the one with the smallest effect. The effect of each level is computed as follow: it is the sum of the considered measurement (length of returned solution or optimization time) of all the experiments for which the factor is set to the corresponding level. Then, the effect of each level is normalized (by dividing it by the mean of the considered measurement of all the experiments of the DoE) to be able to compare the tendencies over all the test cases.

For example, one can consider the DoE carried out on the Stem part. For this DoE, 32 experiments have been repeated 5 times. For each experiment, the repetitions with the lowest and the highest lengths have been discarded, and the trimmed length response of the experiment has been computed as the sum of the length of the 3 remaining repetitions divided by 3. Then, the trimmed length response of all the experiments have been gathered in a table. To compute for instance the effect of the Elitist level for the Selection operator, the experiments for which the Selection operator is set to Elitist are considered and the corresponding effect is computed as the mean of these experiments trimmed length responses. Finally, this effect is divided by the mean of all the trimmed length responses of the Stem DoE, giving 110.7% according to figure 8.

For the length measurement (fig. 8), it can be noted that the tendencies of the effects are globally similar over the three parts. For the selection, crossover and reinsertion operators, the optimal levels are identical for the 3 test cases (namely the Tournament, Uniform and Elitist levels). For the mutation operator, the Reverse and the Twors levels seem more effective than the Uniform level. However, between the two, none is better than the other on all the test cases. Therefore, the Twors level has been arbitrarily selected. For the MinSize parameter, the two highest levels (namely 500 and 1000) seem to be more efficient than the others, but the best one is difficult to isolate. The same tendency can be noticed for the Crossover Probability (with the 0.75 and 1 levels) and for the Mutation Probability (with the 0.3 and 0.5 levels). For the Span factor, the 500 level is better than the others for the Turbina and the Stanford Bunny parts, but it returns a slightly longer solution for the Stem part.

For the proposed framework, the length measurement can be seen of more importance in comparison to the optimization time, because the latter should be negligible with regards to the production time of the parts (especially in the case of series production). Therefore, to select the best parameters for the proposed framework, the most beneficial levels over the length measurement have been selected, and for the conflicting parameters, the level with the lowest time-consumption has been chosen. Following this rule, the 8 optimized parameters values are presented in table 2. Thus, for the mutation operator, the time consumption cannot help to decide between the Reverse and the Twors level. However, for the MinSize parameter, the 1000 level clearly increases the optimization time, and the 500 level will be favored. Likewise, for the Crossover Probability, the 0.75 level will be privileged

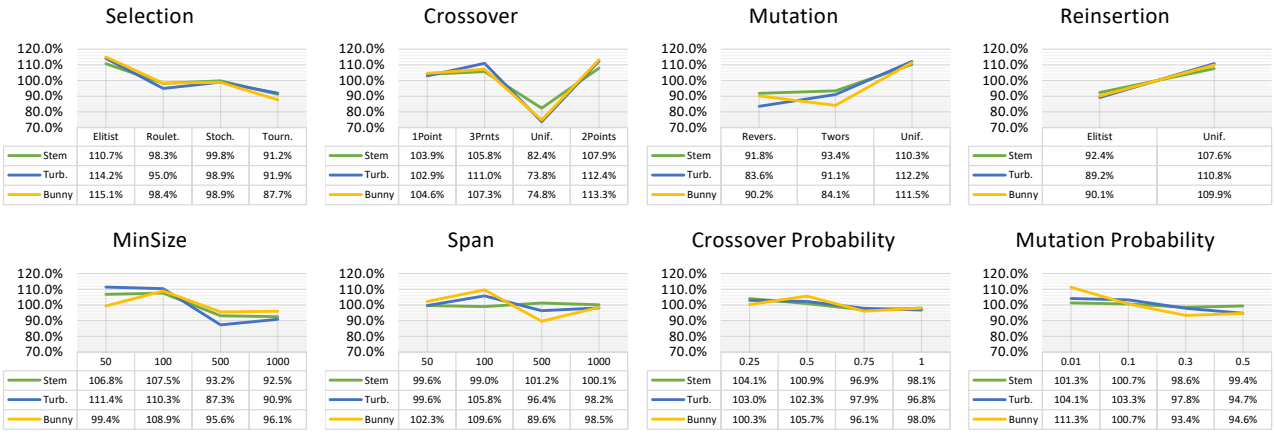

Figure 8: Effect analysis of the DoE regarding the length of the solution

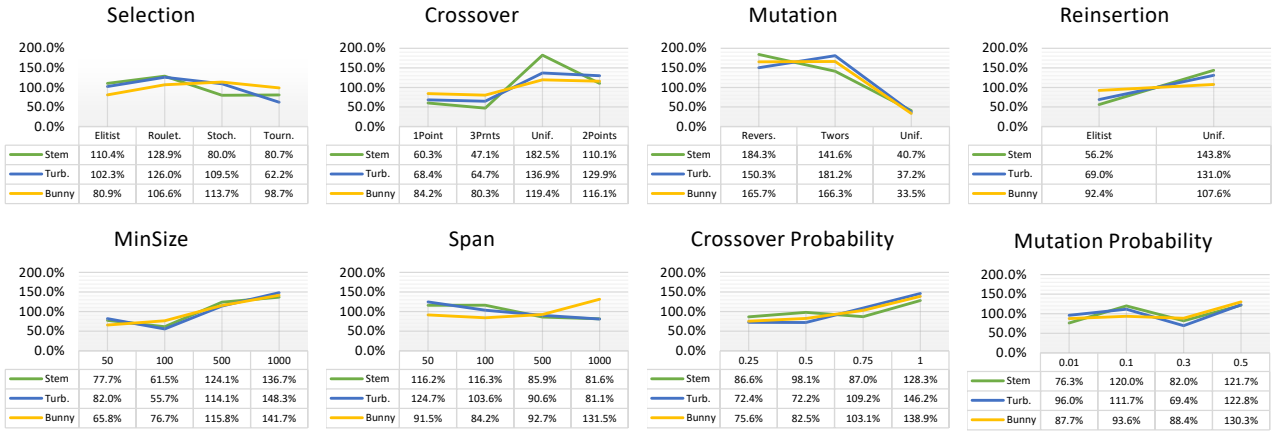

Figure 9: Effect analysis of the DoE regarding the optimization time

over the 1 level, and for the Mutation Probability, the 0.3 level will be chosen over the 0.5 level. For the Span parameter, the time-consumption graph comfort the preselection made.

Parameters	Optimized levels
Selection	Tournament
Crossover	Uniform
Mutation	Twors
Reinsertion	Elitist
MinSize	500
Span (MaxSize-MinSize)	500
Crossover Probability	0.75
Mutation Probability	0.3

Table 2: Optimized parameters levels implemented for the proposed framework.

5.2. *Lattice parameters optimization*

As mentioned in section 3, the proposed framework does not consider the parameters of the initial lattice as variables of the optimization process. Thus, those parameters have to be tuned before generating the initial lattice structure and before solving the LS^{2}DO problem using the GA. The lattice structure parameters are as follows (fig. 4):

- maximal beam angle α_{max} which corresponds to the maximal angle between a beam of the lattice and the horizontal plane. It can vary between 45° and 90°.
- beam diameter d which value is constrained by the adopted technology. For LBM technology, it can vary between 0.5mm (minimal beam diameter that can be manufactured) and $+\infty$.
- unit cell parameter a corresponding to the repetition distance of the unit cells in the X and Y directions. It is a lattice generation parameter, but it is not really a lever for action. Indeed, the value of a must satisfy the sustainment condition stated by equation (2). This equation makes a dependent of the beam diameter d and overhanging distance o_ℓ, the two latters being independent from each other. o_ℓ is therefore the true lever of action, and has to be smaller than 1mm ($o_\ell \leqslant 2o_p$) for the LBM technology.

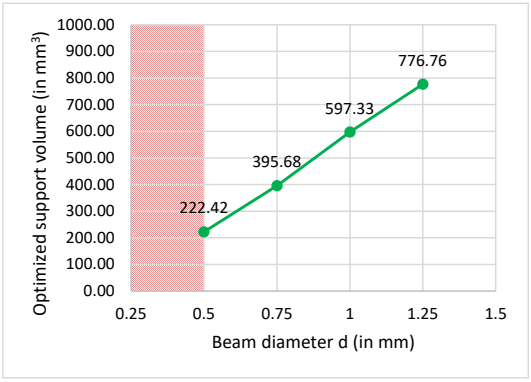

Figure 10: Volumes of optimized internal supports for the Stanford Bunny according to the initial lattice beams diameter d. (unmanufacturable area in red)

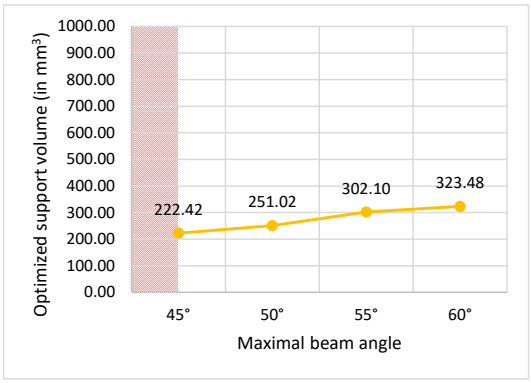

Figure 11: Volumes of optimized internal supports for the Stanford Bunny according to the initial lattice maximal beam angle α_{max}. (unmanufacturable area in red)

According to equation (2), if the overhang distance o_ℓ decreases, the unit cell parameter a decreases and the lattice structure becomes denser. The main objective of the developed framework being to minimize the volume of the lattice support structure, one can clearly understand that the optimal value for o_ℓ is the highest possible.

However, for the beam diameter, increasing the value of d results in a sparser lattice structure, but with thicker beams. Likewise for the maximal beam angle α_{max}, changing the value will result in a lattice with less but longer beams. Therefore, it is not obvious that setting these initial lattice parameters to there lowest values will minimize the optimized support volume. To clarify this point, some experimentations have been carried out by varying the beam diameter d and the maximal beam angle α_{max} on the internal support structure of the Stanford Bunny part. Figure 10 and 11 presents the results of these experimentations. As it can be seen in figure 10, a low beam diameter d induces the lowest volume of the optimized supports. Likewise, on figure 11, a low maximal beam angle produces the lowest volume for the optimized supports. Therefore, the best values for the d and α_{max} parameters of the initial lattice are the lowest possible.

Following those rules, the values of the three control parameters have been chosen equal to the commonly used lower limits of the manufacturing constraints of the LBM technology: $\alpha_{max} = 45°$, $d = 0.5mm$ and $o_\ell = 1mm$. This technology is the one used to print the generated support structures (fig. 15).

5.3. Results comparison

*** TO BE IMPROVED *** To evaluate the interest of using lattice structures to support overhanging area on a part, Table 3 compare the volume of the support structures obtained by using the proposed framework to the volume of the support structures suggested by common commercial software. This comparison is done on the internal and external support structures of the 3 test cases.

Table 3 clearly shows that the proposed framework enables the generation of the support structures with the lowest volume. However, the supports generated by commercial software strategies are generic ones and could be optimized furthermore. The conclusion to be drawn from Table 3 is thus not that the proposed framework is better than any commercial solution available, but rather that it generates low volume support structures in comparison to what is generally used in the industry.

6. Conclusions and future works

In this paper, a new framework has been proposed to optimize support structures for additive manufacturing. The aim of the framework is to sustain all the overhanging areas of a part, leaving aside the deformation and thermo-accumulation issues. To do so, a manufacturable lattice structure is generated under the overhanging areas. Then, a genetic algorithm optimizes this lattice by removing the maximum number of beams, while insuring that all the areas to support are still sustained.

This article has presented the various results in the utilization of this framework: the parameters of the GA have been optimized through a design of experiments, the internal and external support structures of 3 test cases have been successfully generated by the proposed framework and manufactured, and their volumes have been compared to the ones of support structures suggested by commercial software, underlining the interest of the developed algorithm in terms of volume optimization.

As a perspective for this research, the initial population selection of the genetic algorithm could be further improved through the implementation of heuristic search. The convergence of the GA itself could also be increased thanks to a quick local search done after each crossover and mutation, in order to obtain better child chromosomes.

Furthermore, because the deformation and thermal accumulation problems have been left aside, the proposed framework cannot be used to support any kind of additively manufactured part. However, it is a first

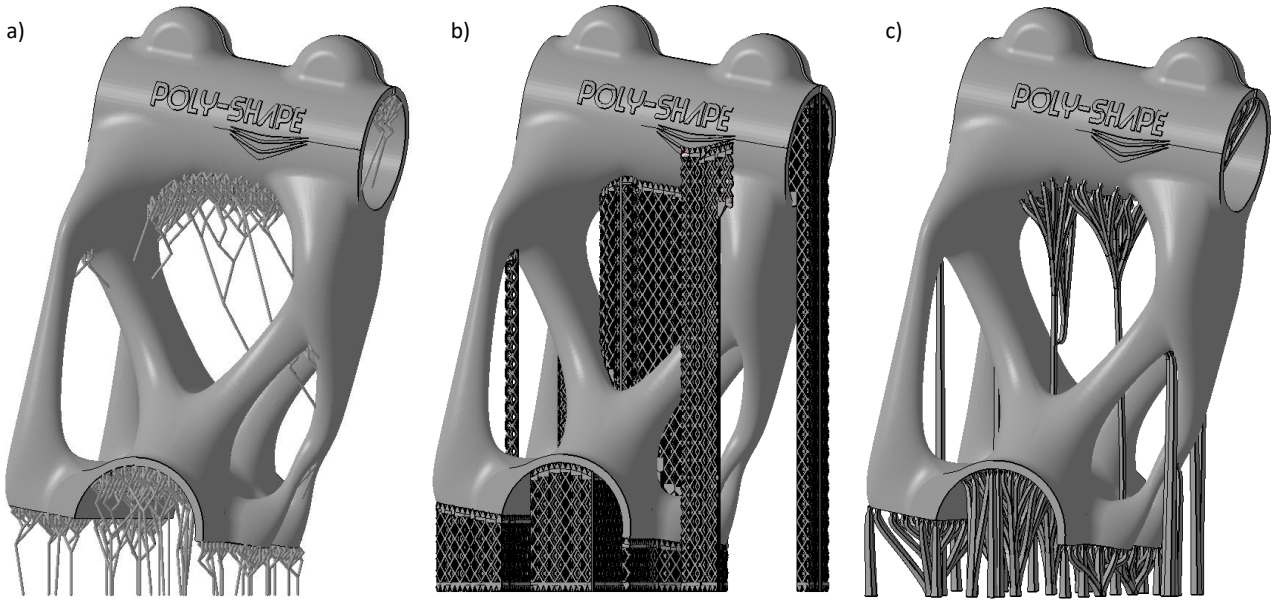

Figure 12: Stem external supports generated by our framework (a), by the SLM support strategy (b) and the DLP support strategy (c) of commercial software.

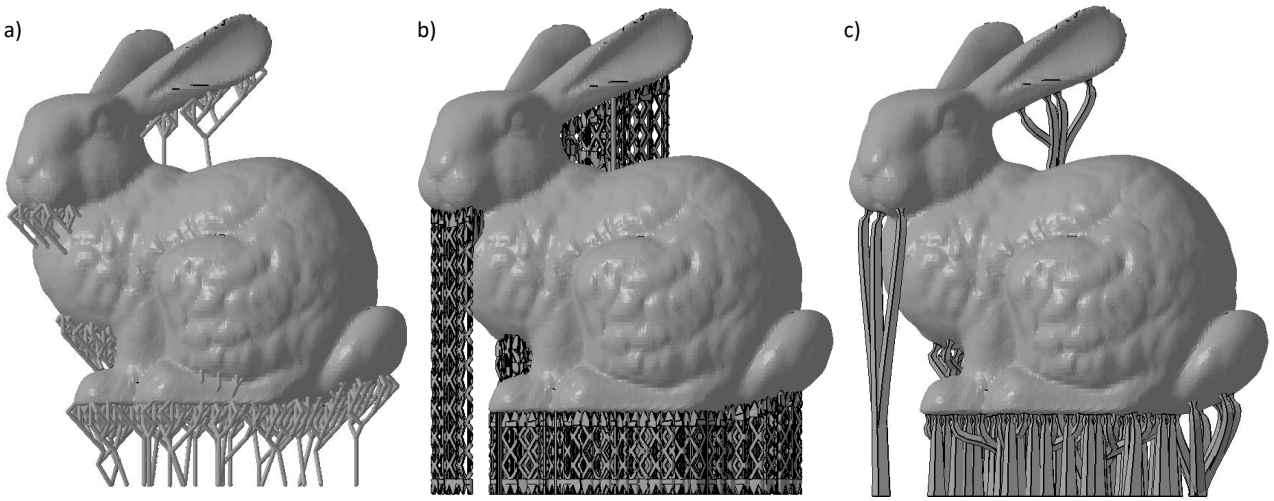

Figure 13: Standford Bunny external supports generated by our framework (a), by the SLM support strategy (b) and the DLP support strategy (c) of commercial software.

	Stem supports		Turbine supports		Bunny supports	
	internal	external	internal	external	internal	external
	Volumes (in mm^3)					
LS^2DO using GA	420	1300	45	1465	210	320
(Commercial) SLM strategy	630	3100	–	1790	380	470
(Commercial) DLP strategy	620	3880	80	–	390	1720
	Gain					
Gain LS^2DO wrt SLM	-33.3%	-%	–	+xx%	-%	-%
Gain LS^2DO wrt DLP	+xx%	-66.5%	+xx%	–	-46.2%	-81.4%

Table 3: Volumes comparison between solutions obtained by the proposed framework and the support structures suggested by commercial software.

block in the wide area of support structure optimization. Its coupling with thermo-mechanical optimization algorithms is of interest in the future, in order to generate poly-functional support structures, that can sustain overhangs, rigidify features subject to deformation, and dissipate the thermal accumulation areas of any additively

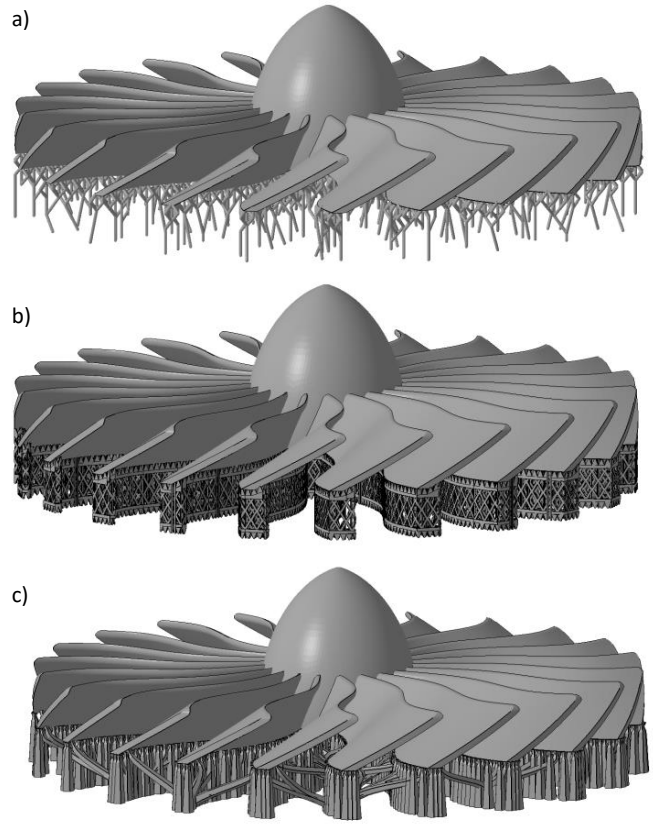

Figure 14: Turbina external supports generated by our framework (a), by the SLM support strategy (b) and the DLP support strategy (c) of commercial software.

Figure 15: The Stem test case printed in LBM

manufactured part.

References

[1] S. A. Tofail, E. P. Koumoulos, A. Bandyopadhyay, S. Bose, L. O'Donoghue, C. Charitidis, Additive manufacturing: scientific and technological challenges, market uptake and opportunities, Materials Today.

[2] U. M. Dilberoglu, B. Gharehpapagh, U. Yaman, M. Dolen, The Role of Additive Manufacturing in the Era of Industry 4.0, Procedia Manufacturing 11 (2017) 545–554.

[3] ASTM F2792-12a, Standard Terminology for Additive Manufacturing Technologies, (Withdrawn 2015) (2012).

[4] A. Hussein, L. Hao, C. Yan, R. Everson, P. Young, Advanced lattice support structures for metal additive manufacturing, Journal of Materials Processing Technology 213 (7) (2013) 1019–1026.

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

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

[7] J. Jhabvala, E. Boillat, C. André, R. Glardon, 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 (1-4) (2012) 137–142.

[8] T. A. Krol, M. F. Zaeh, J. Schilp, C. Seidel, Computational-Efficient Design of Support Structures and Material Modeling for Metal-Based Additive Manufacturing.pdf.

[9] T. A. Krol, M. F. Zaeh, C. Seidel, Optimization of Supports in Metal-Based Additive Manufacturing by Means of Finite Element Models.

[10] Y.-a. Jin, Y. He, J.-z. Fu, Support generation for additive manufacturing based on sliced data, The International Journal of Advanced Manufacturing Technology 80 (9-12) (2015) 2041–2052.

[11] S. S. Crump, J. W. Comb, W. R. Priedeman Jr, R. L. Zinniel, Process of support removal for fused deposition modeling, uS Patent 5,503,785 (Apr. 1996).

[12] B. Qian, Z. Lichao, S. Yusheng, L. Guocheng, Support fast generation algorithm Based on discrete-marking in rapid prototyping, Affective Computing and Intelligent Interaction (2012) 683–695.

[13] B. Cheng, K. Chou, Geometric consideration of support structures in part overhang fabrications by electron beam additive manufacturing, Computer-Aided Design 69 (2015) 102–111.

[14] B. Cheng, Y. K. Chou, Overhang Support Structure Design for Electron Beam Additive Manufacturing, ASME, 2017, p. V002T01A018.

[15] K. Cooper, P. Steele, B. Cheng, K. Chou, Contact-Free Support Structures for Part Overhangs in Powder-Bed Metal Additive Manufacturing (2015) 12.

[16] M. Gan, C. Wong, Practical support structures for selective laser melting, Journal of Materials Processing Technology 238 (2016) 474–484.

[17] Boyard N., Méthodologie de conception pour la réalisation de pièces en Fabrication Additive, Ph.D. thesis (2015).

[18] Hussein A., The Development of Lightweight Cellular Structures for Metal Additive Manufacturing.pdf, Ph.D. thesis (2013).

[19] M. Cloots, A. B. Spierings, K. Wegener, Assessing new support minimizing strategies for the additive manufacturing technology SLM, in: 24th International SFF Symposium-An Additive Manufacturing Conference, Austin, USA, University of Texas at Austin, 2013, pp. 631–643.

[20] D. Li, N. Dai, X. Jiang, Z. Shen, X. Chen, Density Aware Internal Supporting Structure Modeling of 3d Printed Objects, IEEE, 2015, pp. 209–215.

[21] J. Lee, K. Lee, Block-based inner support structure generation algorithm for 3d printing using fused deposition modeling, The International Journal of Advanced Manufacturing Technology.

[22] B. Swaelens, J. Pauwels, W. Vancraen, Method for supporting an object made by means of stereolithography or another rapid prototype production method, uS Patent 5,595,703 (Jan. 1997).

[23] J. Vanek, J. A. G. Galicia, B. Benes, Clever Support: Efficient Support Structure Generation for Digital Fabrication, Computer Graphics Forum 33 (5) (2014) 117–125.

[24] X. Zhang, Y. Xia, J. Wang, Z. Yang, C. Tu, W. Wang, Medial axis tree—an internal supporting structure for 3d printing, Computer Aided Geometric Design 35-36 (2015) 149–162.

[25] R. Vaidya, S. Anand, Optimum Support Structure Generation for

Additive Manufacturing Using Unit Cell Structures and Support Removal Constraint, Procedia Manufacturing 5 (2016) 1043–1059.
[26] L. Chougrani, J.-P. Pernot, P. Véron, S. Abed, Lattice structure lightweight triangulation for additive manufacturing, Computer-Aided Design 90 (2017) 95–104.
[27] M. Hauptmann, M. Karpiński, A compendium on Steiner tree problems, Inst. für Informatik, 2013.
[28] D. Watel, Approximation de l'arborescence de Steiner, Ph.D. thesis, Versailles-St Quentin en Yvelines (2014).
[29] E. Halperin, R. Krauthgamer, Polylogarithmic inapproximability, in: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing, ACM, 2003, pp. 585–594.
[30] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, M. Li, Approximation Algorithms for Directed Steiner Problems.
[31] A. Zelikovsky, A series of approximation algorithms for the acyclic directed Steiner tree problem, Algorithmica 18 (1) (1997) 99–110.