Genetic-algorithm based framework for lattice support structure optimization in additive manufacturing

Benjamin Vaissier, Jean-Philippe Pernot, Laurent Chougrani, Philippe Veron

Genetic-algorithm based framework for lattice support structure optimization in additive manufacturing☆

Benjamin Vaissier [a,b], Jean-Philippe Pernot [a,*], Laurent Chougrani [b], Philippe Véron [a]

[a] Arts et Métiers, LISPEN EA 7515, HeSam, Aix-en-Provence, France
[b] Poly-Shape, 235 rue des Canesteu, Salon-de-Provence, France

Keywords:
Support structures
Lattice structures
Genetic algorithm
Directed Steiner tree
Additive manufacturing

A B S T R A C T

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 most suited geometry for the initial lattice structure is defined at the beginning considering the possibilities of the manufacturing technologies. Then, the pruning of the structure is performed through a genetic algorithm, for which the control parameters values have been tuned through a design of experiments. The proposed approach is validated on several test cases of various geometries, containing both inner and outer areas to be supported. The generated support structures are compared to the ones obtained by several state-of-the-art support structure strategies and are proved to have lower material consumption.

1. Introduction

Additive Manufacturing (AM) has taken a huge step towards industrialization over the last few years, and the field 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 technologies, 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 [4], parts with intricate flow channels [5] 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. Indeed, with most AM technologies, the addition of support structures is required to ensure the good production of a part. Support structures can fulfill three main functions [6]: (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 with the horizontal plane inferior to a threshold value, called overhang angle, usually considered equal to 45° [7]. 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 primarily focuses on the sustainment function, i.e. how overhanging areas can be sustained. The stiffening and dissipation requirements are not directly tackled, but are discussed in the conclusion. Support structures can be classified into two main categories:

- *removable* support structures (also referred to as *external* or *outer* 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. They are also known as *internal* or *inner* support structures. This category is avoided as much as possible.

☆ This paper has been recommended for acceptance by Jun Wu, Xiaoping Qian, and Michael Yu Wang.
* Corresponding author.
E-mail address: jean-philippe.pernot@ensam.eu (J.-P. Pernot).

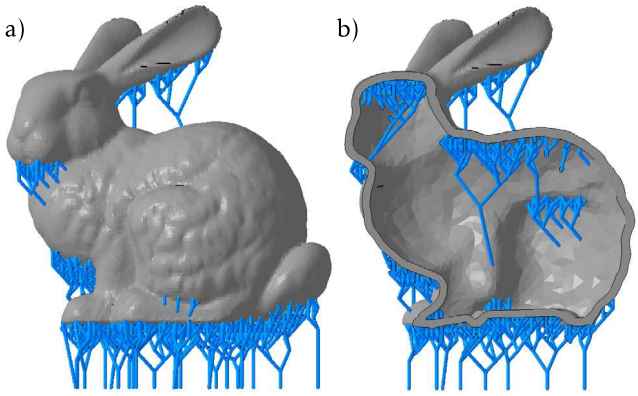

Fig. 1. Lattice support structures generated from a triangle mesh of the Stanford Bunny using the proposed framework: full view (a) and cross-sectional view (b).

To minimize the need of support structures, some approaches focus on generating self-supporting geometries [8]. However, in most practical cases, it is not possible to eliminate all the overhanging surfaces. The optimization of support structures thus 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 diminishes therefore the overall cost of the part. For permanent support structures, by definition, only the volume and the production time are to be minimized.

This article proposes a new framework for the optimization of support structures, based on a discrete optimization of lattice 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 overall volume of the external and internal supports (Fig. 1). The most suited geometry for the initial lattice is defined at the beginning considering the possibilities of the manufacturing technologies. Its topology is then optimized by pruning the lattice through a genetic algorithm. Post-processing steps are finally performed to prepare 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 privileged direction, making it optimal for future mechanical optimization and easily removable during the finishing step; (iii) the framework implements a new methodology using a genetic algorithm to find a solution to the Directed Steiner Tree (DST) problem associated to the underlying lattice support structure optimization.

The paper is organized as follows. After an overview of the current developments on support optimization (Section 2), Section 3 describes the proposed framework composed of several steps. The problem of finding a lightweight 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 best control parameters are selected for each step according to the results of several experimentations (Section 5). The results are compared with the ones obtained by several state-of-the-art support structure strategies. 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 from collapsing and reduce part deformation. However, they represent an important proportion of the production cost (e.g. material volume, production time and support removal time) and optimizing them is therefore essential.

Depending on the adopted technology, 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 into 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 [9]. Järvinen et al. optimize part surface roughness and removability of tube and web extruded structures by varying various parameters such as the thickness [10]. Jhabvala et al. generate filled block support structures with porous micro-structure, thanks to a pulsed laser, making them easily removable [11]. Krol et al. developed a discrete optimization based on a Finite Element Analysis (FEA) model by subdividing extruded crossing walls [12,13]. For the Fused Deposition Modeling (FDM) technology, Jin et al. propose a slice-based algorithm to generate extruded plastic support structures [14] and Huang et al. extend this method with a pixel-based algorithm [15,16]. 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 [17]. 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 [18]. 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 [19–21].

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 [22]. In FDM, Boyard repeats a tree-like structure under the overhanging areas to support plastic parts [23].

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 [6,24], whereas Cloots et al. stack parts on top of each other in the building chamber by using lattice support structures [25]. In FDM, Li et al. vary the diameter of lattice structure beams in order to create stiffer support structures [26], whereas Lee et al. propose a voxel-based hollowing method to create inner support structures [27]. In SLA, Swaelens et al. filed a patent on the geometries of support structures, including perforated crossing walls and lattice support structures [28].

The *aperiodic* support structure category gathers all the support geometries that do not present any repetition pattern. For example in FDM, Mezzadri et al. generate organic support structures through a topology optimization of the support volume. On their side, Schimdt et al. have developed an algorithm to generate tree-like support structures [29]. This algorithm has latterly been implemented in the free software Meshmixer, and will be compared to the framework proposed in this article. Vanek et al. also

generate tree-shaped support geometries by computing the intersection of cones placed under overhanging surfaces [30] whereas Vaidya et al. repeat octahedral unit cells to achieve the same goal [31]. Dumas et al. and Shen et al. also use the possibility of FDM to create bridges (large overhanging areas sustained at its two ends) to reduce support volume [32,33]. To support internal cavities, Mao et al. propose a hybrid approach combining muscle fiber inspired structures and triply periodic structures to optimize locally the strength-to-weight ratio of the part. Stava et al. also support cavities by adding inner structures, and propose a tetrahedron-based hollowing process to reduce the mechanical loads on supports [34]. Finally, the internal support structures can also be compared with the works on part hollowing of Wu et al. [35], Lee et al. [36] Xie et al. [37].

Most of the previously mentioned works propose algorithms to support all the overhanging surfaces but few take into account the importance of minimizing the volume of the generated structures. Besides removing unnecessary beams and thus unnecessary fused material, our framework operates on a pre-optimized lattice structure whose geometry has the minimal volume to make it manufacturable. In Section 5, the results generated by the proposed framework are compared to the ones obtained by some of the previously introduced state-of-the-art approaches.

Furthermore, many approaches are exploring extruded, dually periodic and triply periodic structures but few focus 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 optimizes a lattice structure by removing the unnecessary beams and therefore generates aperiodic tree-like structures, presenting no repetition bias. Finally, working on such a tree-like structure also contributes to ease the removal of the external supports during the finishing step.

3. Overall framework

This section introduces the new optimization framework developed to sustain part overhanging surfaces. As a reminder, an overhang corresponds to a surface forming an angle usually considered inferior to 45° with the horizontal plane [7].

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 (with p referring to process) from at least one other point $\mathbf{p}_1$ with support material directly below it ($\mathbf{p}_1$ is therefore necessarily on the border of the overhanging surface). The overhang distance o_p depends on the adopted AM technology, material and print parameters. For example, with the FDM technology, the overhang distance o_p can reach dozen of centimetres [32] (depending on the material) whereas with the LBM technology, it is considered that $o_p \simeq 0.5$ mm. This is illustrated in Fig. 2 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 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 means 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 first generate such an initial lattice structure under the overhanging surfaces of a part, and then to remove from this lattice the maximum number of beams, without however breaking the sustainment condition. More precisely, the proposed framework is composed of several steps illustrated in Fig. 3:

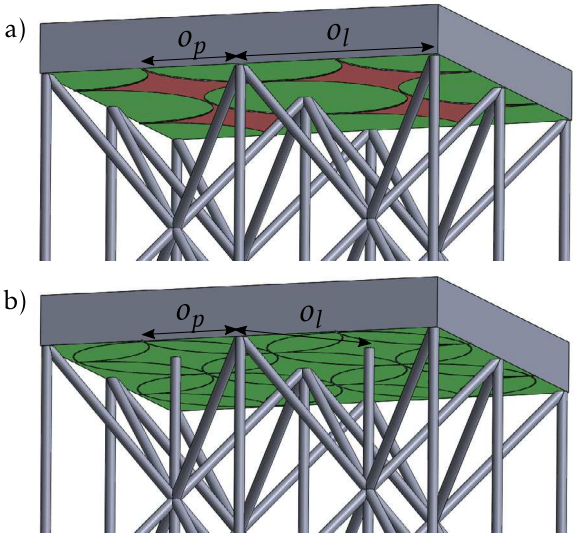

Fig. 2. The sustainment condition: a partially (a) and a fully (b) sustained surface. The unsustained areas are in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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 three 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 five 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 α_{max} and the overhanging distance of the lattice o_ℓ (with ℓ referring to lattice) can be easily computed. This lattice must satisfy the sustainment condition (1) because the optimization algorithm identifies the best sublattice of this initial lattice. Thus, the sustainment condition for this specific unit cell is $o_\ell \leqslant 2o_p$ which gives:
   $$a \leqslant (d + 2o_p)\sqrt{2} \quad (2)$$
   The lattice is then trimmed to the part surface so that every beam going through the part surface is cut short. Finally, the parameters of the lattice are not considered as variables of our lattice support structure discrete optimization. Thus, the best values are selected 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

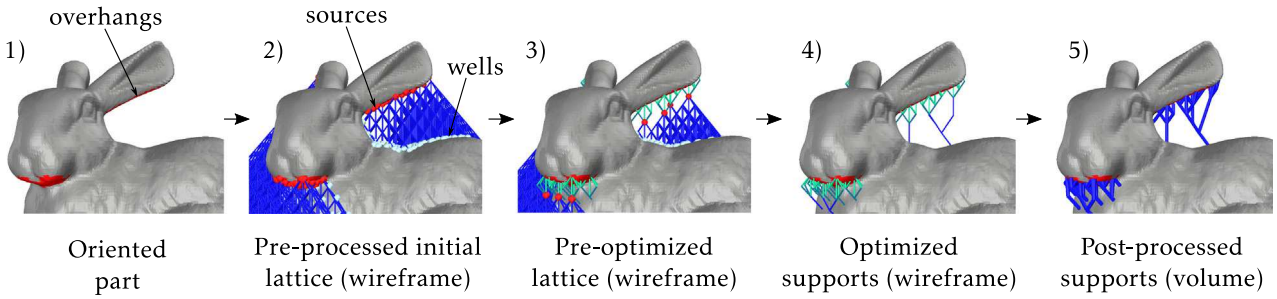

Fig. 3. The proposed optimization framework (example on the Standford Bunny external supports).

overhanging area are called sources and are the ones that must be sustained. 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 directly deduced are identified and removed from the set of optimization variables, thus reducing the computation time. For example, if a source node has only one outgoing beam, this beam is needed in order 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) attempts to determine which sublattice 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 termination conditions are reached. This algorithm is introduced in Section 4 and the method used to tune its control parameters is discussed in Section 5.1.
5. *Post-processing*: once the sublattice is found (in the finite solution set defined by the lattice structure), a post-processing step is applied to even further reduce its length (in a different solution set, spatially continuous). The beam paths between connection points (i.e. sources, wells and points of valency greater than 2) 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, and will not generate any new overhanging area. Then, the resulting lattice structure is triangulated using for instance the approach of Chougrani et al. [38] which minimizes the number of generated triangles before printing.

At the end of this optimization process, some preparation steps are still required before printing. For example, the generated geometry is to 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 to find a solution to this problem, a Directed Acyclic Graph (DAG) is associated with the initial lattice and a Genetic Algorithm (GA) is used to find the subgraph respecting the sustainment constraints and with the smallest cumulated beam length.

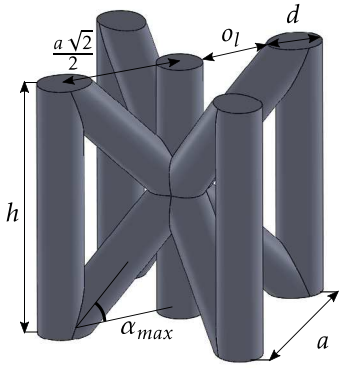

Fig. 4. The unit cell used in this paper.

4.1. Problem formalization

In this paper, a unique lattice graph $G = (V, E)$ is associated to a lattice L (defined as a set of interconnected beams) through the following construction process. To each beam extremity of L is associated a unique vertex $v \in V$ (a junction node between several beams being considered as the same unique extremity). Then, to each beam of L is associated a unique edge $e \in E$ connecting the two vertices in V corresponding to the beam extremities. The unicity of the lattice graph associated to a specific lattice can clearly be deduced from the construction process. To each subgraph G' of G, a unique lattice can be constructed by removing all the beams of L for which the corresponding edge of G is not in G'. Such a lattice is called a sublattice of L.

An element of V is denoted v_i with $i \in \{1, \ldots, |V|\}$. 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 note out_{max} is the maximum number of outgoing beams in the graph, i.e. $out_{max} = max_{i=1..|V|}\{|out(v_i)|\}$. Let us also note V_{out} is the set of all vertices in V with at least one outgoing edge: $V_{out} = \{v_i \in V : out(v_i) \neq \emptyset\}$. Because each edge is necessarily an outgoing edge for one of its extremity vertex, the number of beams of G can be written as:

$$|E| = \sum_{v \in V_{out}} |out(v)| \tag{3}$$

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

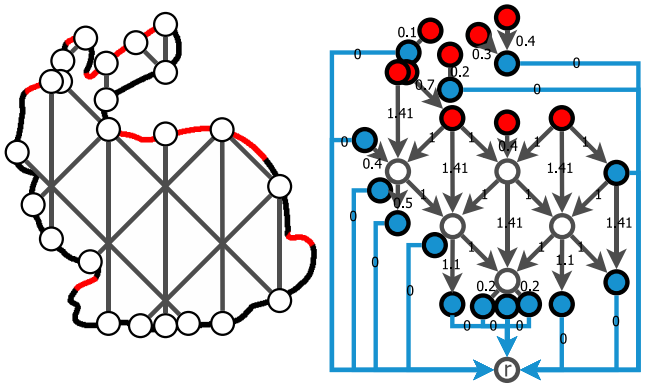

Fig. 5. Example of LS^{2}DO problem on a 2D Stanford Bunny (left) and the weighted directed acyclic graph (DAG) associated (right). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

lattice graph a cost value equal to the length of the corresponding lattice beam. In this paper, the cost value $w_e(e_j)$ of an edge e_j, with $j \in \{1, \ldots, |E|\}$, corresponds to the length of the corresponding beam in the lattice. Consequently, the weight (or length in the present case) $w(G)$ of a lattice graph $G = (V, E)$ is given by the sum of all the costs of its edges.

The framework starts by generating a lattice structure L under the overhanging areas (red borders on the 2D Stanford Bunny of Fig. 5) and optimizes it by finding a sublattice of L with the minimum cumulated beam length, that sustains all the identified sources in V_S. In terms of graph, the problem is to find a subgraph G' 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. The optimization problem (associated to the LS^{2}DO problem) can therefore be defined as follows:
Minimize:

$$w(G') = \sum_{j=1}^{|E'|} w_e(e_j) \tag{4a}$$

subject to:

$$G' = (V', E') \in sub(G) \text{ with } V_S \subset V' \tag{4b}$$

$$\forall s \in V_S, \exists w \in V_W \text{ s.t. } w \in V' \text{ and } \mathcal{P}_{G'}(s, w) \neq \emptyset \tag{4c}$$

where $sub(G)$ defines the set of all subgraphs of G and $\mathcal{P}_{G'}(s, w)$ the set of all paths from s to w in G'. Fig. 5 shows the weighted DAG associated to the LS^{2}DO problem (in gray). Red vertices are the sources, and blue vertices are the wells.

Now, if one connects all the well vertices to a new root vertex r, by simply adding a set of edges $\widetilde{E}$ weighted with a 0 value (blue in Fig. 5), the problem remains unchanged, but it can be formulated as: "Find the tree $T = (V_T, E_T)$ in $G_{ini} = (V \cup \{r\}, E \cup \widetilde{E}, w_e)$ with the minimal cost $w(T)$, rooted at r such that $V_S \subset V_T \subseteq V \cup \{r\}$ and $E_T \subseteq E \cup \widetilde{E}$". Therefore in Fig. 5 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.

One could notice the similarity between this problem and the classical Spanning Tree (SpT) problem. However, this problem differentiates itself from the rooted and directed version of the SpT problem since only a subset of the graph vertices must be connected to the root. Actually, the previous formulation corresponds to the definition of the Directed Steiner Tree (DST) problem known to be NP-hard [39,40]. Halperin et al. proved that, for any $\epsilon > 0$, there is no polynomial approximation algorithm within a $\log^{2-\epsilon} n$ ratio, unless $P = NP$ [41]. Thus, many subjects of research focus on approximating the solution to the DST problem. For example, Charikar et al. present an algorithm with an approximation ratio of $O(k^{2/3} \log^{1/3} k)$, where k is the number of pairs of vertices to be connected [42]. Zelikovsky also proposes an $O(k^{\epsilon})$ -approximation algorithm for any $\epsilon > 0$ in the case of a DAG [43].

4.2. Resolution using a genetic algorithm

The resolution of the LS^{2}DO problem is performed using a Genetic Algorithm (GA) whose control parameters have been tuned through a design of experiments. Genetic algorithms are metaheuristics. They are generic algorithms, adaptable to various kinds of problems, and display a random nature. In this sense they are non-deterministic algorithms. They are part of the larger family of evolutionary algorithms. Actually, genetic algorithms, and metaheuristics in general, have been successfully used in the past and have proved their efficiency to find solutions to variants of Steiner Tree problems [44–46], and to solve complex geometrical or mechanical NP-hard problems [47–49].

A GA manipulates populations of chromosomes. Each chromosome is a set of variables, called genes, encoding a potential solution to a problem, i.e. a subgraph in the present case. The values that can be taken by a particular gene are called alleles. To find an approximate solution to the DST problem associated to 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 subgraph 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 so-called *activation chromosome*. The activation chromosome thus contains $|E|$ genes, each of which has two possible alleles: 0 or 1. If $x_i = 0$, then the edge e_i is not activated, else if $x_i = 1$ it is activated. Every subgraph of a graph can thus be encoded. For the specific LS^{2}DO problem, each subgraph of the initial one is not necessarily a solution of the problem, i.e. there is not necessarily a path from each source to at least a well as defined by Eq. (4c). Therefore, a validation step needs to be executed in order to ensure that Eq. (4c) is satisfied.. Let us note $C_{activation}$ is the set of all the chromosomes defined by the activation encoding. The number of potential solutions thus defined is therefore:

$$|C_{activation}| = 2^{|E|} = 2^{\sum_{v \in V_{out}} |out(v)|} = \prod_{v \in V_{out}} 2^{|out(v)|} \tag{5}$$

- the *switch node encoding* (Fig. 6 right) is another way to encode a solution to the DST problem into a chromosome. Here, a variable x_i is associated to each vertex v_i in the set V_{out} of the initial graph, selecting the only outgoing edge in $out(v_i)$ that will be activated if at least one incoming edge in $in(v_i)$ is activated. The *switch node chromosome* thus contains $|V_{out}|$ genes, and the gene x_i takes values in $\{1, \ldots, |out(v_i)|\}$. Let us note C_{switch} is the set of all the chromosomes defined by the switch node encoding. The number of possible chromosomes define by the switch node encoding is:

$$|C_{switch}| = \prod_{v \in V_{out}} |out(v)| \tag{6}$$

In the 2D example of Fig. 6 (right), a maximum of three values can be assigned to the genes x_i, i.e. $out_{max} = 3$. 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

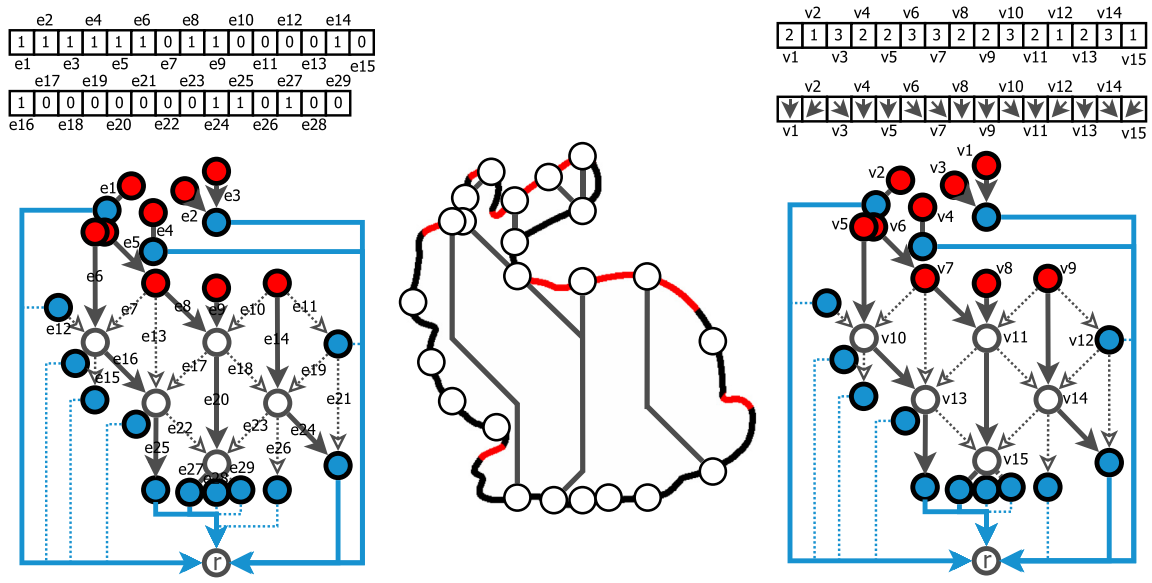

Fig. 6. The activation encoding (left), the switch node encoding (right) and the resulting 2D supports (center).

Table 1
Comparison of the number of chromosomes between the activation and the switch node encoding strategies on the initial lattice structure of the 3D Standford Bunny external supports ($\alpha_{max} = 45°$, $d = 0.5$ mm and $o_\ell = 1$ mm).

	Encoding strategies	
	activation	switch node
Before pre-optimization	10^{25652}	10^{11912}
After pre-optimization	10^{20149}	10^{9357}

only encode subtrees of a graph because for each vertex v_i, only one outgoing edge in $out(v_i)$ can be activated. The set of potential solution is, therefore, the set of all subtrees of the initial graph rooted in r. Actually, since the solution to the DST problem must be a tree rooted in r, the *switch node encoding* is particularly adapted to to find a solution to the LS^{2}DO problem.

By comparing Eqs. (5) and (6), one can notice that the set of all subtrees rooted in r is consistently smaller than the set of all subgraphs. Therefore, the *switch node encoding* reduces greatly the number of considered solutions. Considering a 3D lattice structure composed of basic cells as the one of Fig. 4, the variable x_i associated to a vertex v_i of the initial graph can take 5 different values corresponding to the 5 directions of the oriented edges starting from v_i and going down. Naturally, with another type of unit cell, the values of the variables are to be changed. Table 1 gives an overview of the number of potential chromosomes for each encoding strategy on the initial lattice support structure of the 3D Standford Bunny. Those values are computed from Eqs. (5) and (6) with $|V_{out}| = 17\,043$ before preoptimization, $|V_{out}| = 13\,387$ after preoptimization and such that $\forall v \in V_{out},\ out(v) = 5$.

It clearly reflects the importance of reducing the number of potential solutions by selecting the appropriate encoding strategy and by using a pre-optimization strategy (step 3 of the framework). As a consequence, in this paper, the switch node encoding has been adopted.

The execution of the GA is described in the flowchart of Fig. 7. At first, an initial population of chromosomes is selected. Naturally, the number of chromosomes is much smaller than the ones displayed in Table 1. In this paper, the initial population is chosen randomly but it can be selected through a heuristic algorithm to obtain good initial solutions. Then the so-called *fitness* of each

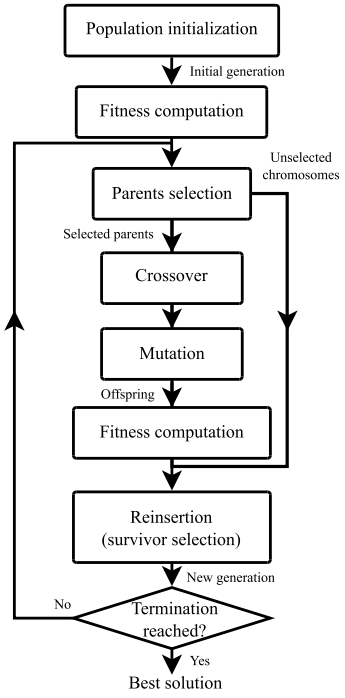

Fig. 7. Genetic algorithm flowchart.

initial chromosome is computed in parallel. The fitness of a chromosome is directly equal to the sum of the weights associated with 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 give birth to the next generation of chromosomes, through crossovers and mutations. 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 the GA. Once the child chromosomes are created, their fitness is once again computed. Some children are then selected to be reinserted into the previous population to form the next generation of chromosomes. Finally, the GA checks if the termination conditions are fulfilled: if so, the best chromosome of the last generation

Table 2
Factors and their levels for the design of experiments parameterization.

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

is considered as the solution to the problem, and otherwise, the evolutionary algorithm goes on with another generation created through the previous steps.

Finally, the GA makes use of 4 operators: selection, crossover, mutation and reinsertion. The GA also 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, the GA is controlled by 8 parameters for which the best values must be selected in order to find a solution to the LS^{2}DO problem as efficiently as possible. The way the 8 control parameters are selected is explained in the next section.

5. Experimentations and results

This section addresses the approach used to select the GA control parameters through a DoE, and the one used to identify the initial lattice structure parameters. The proposed framework is then applied to several test cases and the generated support structures are compared to the ones obtained with several state-of-the-art approaches.

5.1. GA parameters selection

The GA adopted in the proposed framework has been implemented through the GeneticSharp library [50], created by Giacomelli, and available on GitHub. Therefore, the levels for the 4 operators of our GA are the ones of this implementation. Table 2 lists the levels of the 8 parameters (4 operators plus 4 execution parameters) controlling our GA: 6 parameters with 4 levels, 1 parameter with 3 levels and 1 parameter with 2 levels. A brief explanation of each operator possible values is given in the following paragraphs.

Selections: the so-called Elite selection is selecting only the chromosomes with the best fitness values. For the Roulette Wheel selection, the probability of a chromosome to be selected is the ratio of its fitness over the sum of the fitness values of all the chromosomes of the population. The Stochastic Universal Sampling is a variation of the Roulette Wheel selection which ensures that a chromosome with a 3.8% selection probability, for example, is selected in practice 3 or 4 times out of a 100 (and not 0, 1 or 2 times which could happen with uncontrolled randomness). The Tournament selection operator creates random pairs of chromosomes and discards for each pair the chromosome with the lowest fitness value, until the population size fits the requirements.

Crossovers: for the One Point crossover operator, a swapping gene index is randomly defined: all the genes before the swapping index are derived from the first parent whereas the genes after the swapping index are derived from the second parent. The Two Points crossover presents the same mechanism, but with two swapping points. With a Uniform crossover operator, each gene of the child chromosome is independently selected from one parent or the other, according to a mixing probability. For the Three Parents crossover, each gene is also selected independently: if the gene values of the first and second parents are equal, this value is attributed to the child chromosome. Otherwise, the value of the third chromosome gene is attributed.

Mutations: with the Reverse Sequence mutation, a portion of the chromosome is flipped (the first gene of the portion becomes the last and vice-versa). Twors mutation exchanges the position of two randomly chosen genes. With a Uniform mutation operator, the value of a gene is randomly changed, each allele having the same probability to be chosen.

Reinsertions: an Elitist reinsertion is reinserting only the chromosomes with the best fitness values, whereas with a Uniform reinsertion, the offspring are reinserted at random, with the same probability for each chromosome.

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. 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. 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, and the computation time (i.e. time to return a solution). The length of the lattice support structure can then be converted into a volume once the diameter of the beams has been set up.

Because of the random nature of the GA, the DoE can 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 five times, to discard the two extrema (the lowest and the highest measurements) of the two observed quantities, and to analyze the effects of the set of parameters with the mean of the three remaining measurements, called the trimmed response.

Finally, to set up the DoE, it is important to stress that the LS^{2}DO is a part-specific problem: from one part to another, the initial lattice is different, so the evolution of the algorithm may vary. Therefore, the identification of the best GA parameters has been carried out on 3 parts containing more or less complex inner and outer areas to be sustained: the Standford Bunny, an industrial Stem and an industrial Turbine (Figs. 13–15). Those parameters have then been used to run the GA algorithm on other parts which have not been used during the DoE: the Armadillo and Bird (Figs. 16 and 17).

5.1.2. Analysis of the DoE results

Following this DoE, the 32 experimentations have been run 5 times on the 3 parts. Fig. 8 compares the effects of the 3 DoE on the length measurement whereas Fig. 9 compares the effects of the 3 DoE on the time measurement. As a reminder, the length measurement is 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 are ideal for

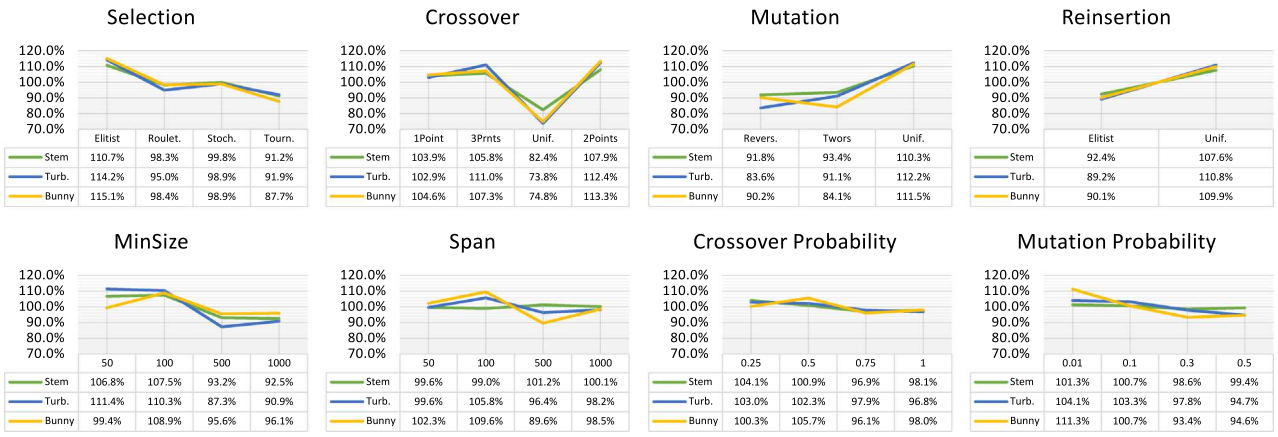

Fig. 8. Effect analysis of the DoE regarding the length of the solution.

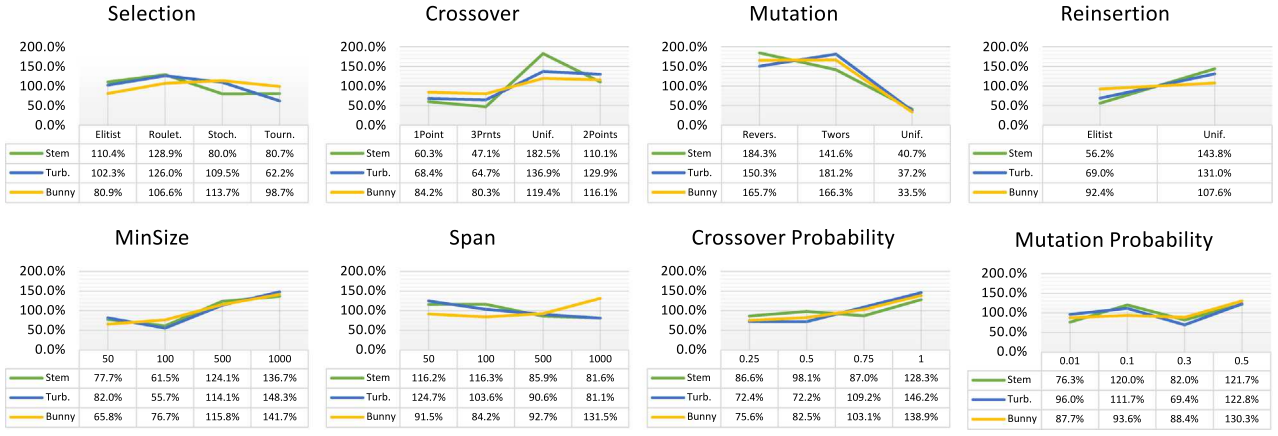

Fig. 9. Effect analysis of the DoE regarding the computation time.

both the length and the computation time. For each parameter, the most suited level is thus the one with the smallest effect. The effect of each level is computed as follows: it is the sum of the considered measurement (length of returned solution or computation 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 has 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 Fig. 8.

For the length measurement (Fig. 8), it can be noted that the tendencies of the effects are globally similar over the 3 parts. For the selection, crossover and reinsertion operators, the ideal 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 one. 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 Turbine and the Stanford Bunny parts, but it returns a slightly longer solution for the Stem part.

For the proposed framework, the length measurement is considered as more important than the computation time, because the latter should be negligible with regard to the production time of the parts (especially in the case of series production). Therefore, to select the best parameters, 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 control parameters values are presented in Table 3. Thus, for the mutation operator, the time consumption cannot help to decide between the Reverse and the Twors levels. However, for the MinSize parameter, the 1000 level clearly increases the computation time, so the 500 level will be favored. Likewise, for the Crossover Probability, the 0.75 level will be privileged 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 comforts the preselection made.

Fig. 10 presents the evolution graph of the LS^{2}DO on the Turbine internal support, using the selected GA parameters levels of Table 3. Each abscissa corresponds to a generation index. The blue continuous line represents the evolution of the best chromosome

Table 3
Selected levels of the GA parameters implemented in our framework.

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

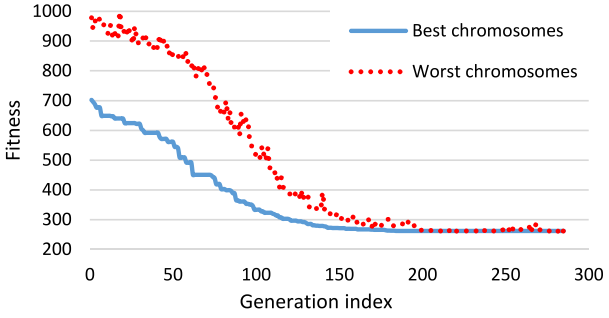

Fig. 10. Evolution graph of the GA on the Turbine internal support.

(with the lowest fitness) of the generation, whereas the red dotted line represents the worst chromosome (with the highest fitness) of the generation. The other chromosomes of the generation are not represented but their fitnesses would lie between these two values. The termination condition of the GA has been set arbitrarily for this test case to 100 generations without evolution of the best chromosome, but this value can be set manually by the user. It can be seen that the fitnesses of the chromosomes of the generations are converging over time to the same low value, indicating that the GA is reaching the end of its search: this converging fitness is probably close (or equal) to the optimal solution length.

5.2. Lattice parameters selection

As mentioned in Section 3, the parameters of the initial lattice are not considered as variables of the optimization process. Thus, those parameters have to be tuned before generating the initial lattice 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 whose value is constrained by the adopted technology. For LBM technology, it can vary between 0.5 mm (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 (Eq. (2)). This equation makes a dependent of the beam diameter d and overhanging distance o_ℓ, the last two being independent from each other. o_ℓ is therefore the true lever of action, and has to be smaller than 1 mm ($o_\ell \leqslant 2o_p$) for the LBM technology.

According to Eq. (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.

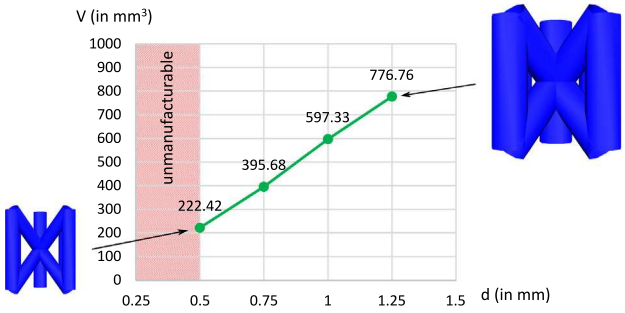

Fig. 11. Volume of internal supports after optimization for the Stanford Bunny according to the initial lattice beams diameter d.

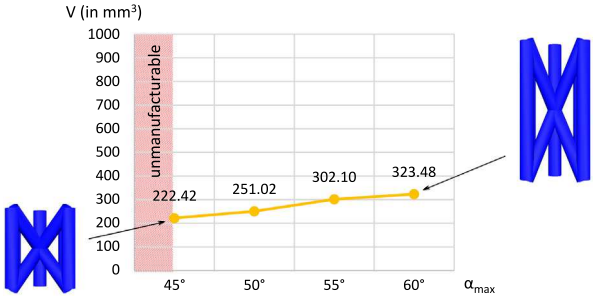

Fig. 12. Volume of internal supports after optimization for the Stanford Bunny according to the initial lattice maximal beam angle α_{max}.

Table 4
Most suitable lattice parameters values implemented in our framework for the LBM technology.

Parameters	Most suitable values
maximal beam angle (α_{max})	45°
beam diameter (d)	0.5 mm
overhanging distance (o_ℓ)	1 mm

However, increasing the value of the beam diameter d results in a sparser lattice structure, but with thicker beams. Likewise, changing the value of the maximal beam angle α_{max} will result in a lattice with less but longer beams. Therefore, it is not obvious that setting these initial lattice parameters to their lowest values will minimize the volume of the lattice support structure solution found by the GA. 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. Figs. 11 and 12 present the results of these experimentations. As it can be seen in Fig. 11, a low beam diameter d induces the lowest volume of the supports after optimization. Likewise, in Fig. 12, a low maximal beam angle produces the lowest volume for the supports after optimization. Therefore, the best values for the d and α_{max} parameters of the initial lattice are the lowest possible.

Following those rules, the three lattice parameters values have been chosen equal to the commonly used lower limits of the manufacturing constraints of the adopted printing technology. For the LBM technology, the adopted parameters are summarized in Table 4. This technology is the one used to print the Stem part of Fig. 18 whose support structure has been generated following the framework proposed in this paper.

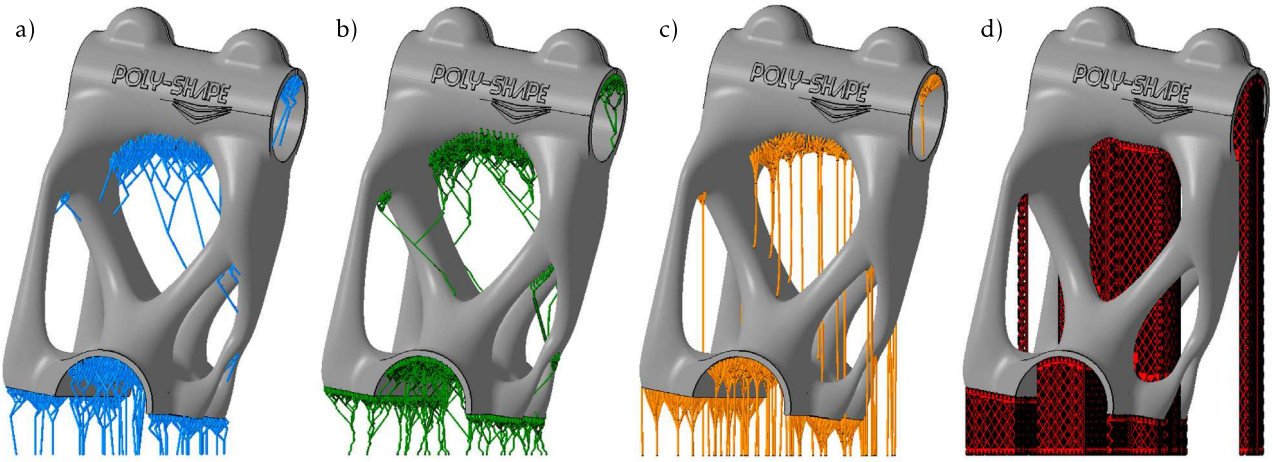

Fig. 13. Stem external supports generated by our framework (a), by Meshmixer (b), by SLA (c) and by SLM (d) support strategies.

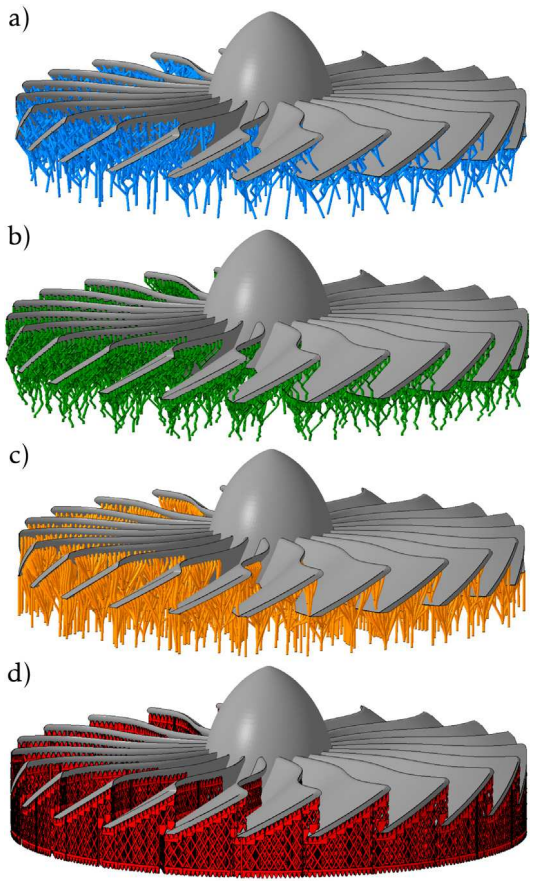

Fig. 14. Turbine external supports generated by our framework (a), by Meshmixer (b), by SLA (c) and by SLM (d) support strategies.

5.3. Results comparison

5.3.1. Volume comparison

To evaluate the interest of using lattice structures to support overhanging areas, the internal support structures obtained by using the proposed framework could be compared to the hollowing methods proposed by [35–37]. However, since these approaches consider other optimization criteria, i.e. taking into account mechanical resistance as well as part manipulation, this comparison would not be fair.

Therefore, a more extended study has been performed to compare the support structure generated by our genetic-algorithm based framework with the support structures obtained by the work of Schmidt et al., implemented in the free software MeshMixer [29] and the ones generated by another software commonly used by industry. For the latter, two supporting strategies are compared to the proposed framework: one dedicated to generating support structures for SLA technology, and the other one for SLM technology. This comparison is done on the internal and external support structures of the 3 test cases (Figs. 13–15). To remove the parts from the building platform, they are positioned at a constant height in order to enable a tool to cut the supports under them. It must be noted that for the comparison, the Meshmixer and the SLA support structures have been generated with the same diameter as the ones generated with the proposed framework ($d = 0.5$ mm).

In addition to the three test cases used to select the GA parameters through the DoE, two other models provided by Vanek et al. [30] have been supported with our framework, with Meshmixer, and with the SLA and the SLM strategies, in order to have an independent comparison: the Standford Amarillo and a model of a Bird. It must be mentioned that the overhang distance o_p used by Vanek et al. is greater than the one used in this article, since the AM technology considered is different (LBM in this article and FDM in their case).

First, with the Meshmixer strategy, it can be noticed that some supports are going around the part (under the chin of the Bunny for example), avoiding to attach the base of the support to the part. Likewise with the SLA support strategy, some support pillars are directly connected to the build platform, whereas a sloping connection to the part itself would decrease their volume. The objective is probably to avoid additional time-consuming part finishing operation, on the junction surfaces. However, the consumption of material, and thus the production time are greatly increased by this choice.

The supports generated with the Meshmixer strategy are also not rectilinear (and present a curly shape), unlike the supports generated with the proposed framework. This is probably due to the optimization strategy executed and must intuitively lead to a degradation of the support mechanical strength.

Contrary to the proposed framework and the Meshmixer strategy, the SLA strategy presents tree-like support structures with vertical trunks: near the overhang surfaces, beams are highly clusterized but after a certain distance, no more beams reunion is found, and the supports finish in a vertical pillar. It is thus less optimized than the proposed framework and the Meshmixer strategy, and the resulting structures have higher volumes.

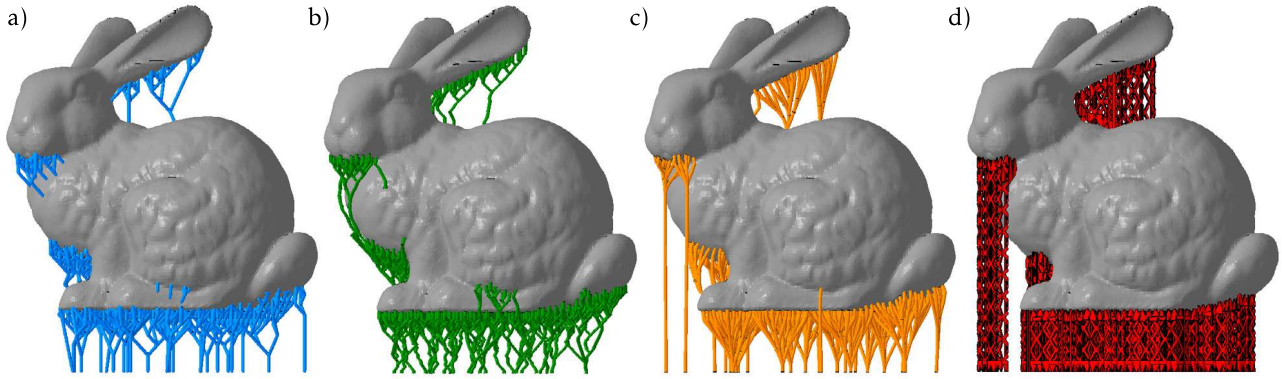

Fig. 15. Standford Bunny external supports generated by our framework (a), by Meshmixer (b), by SLA (c) and by SLM (d) support strategies.

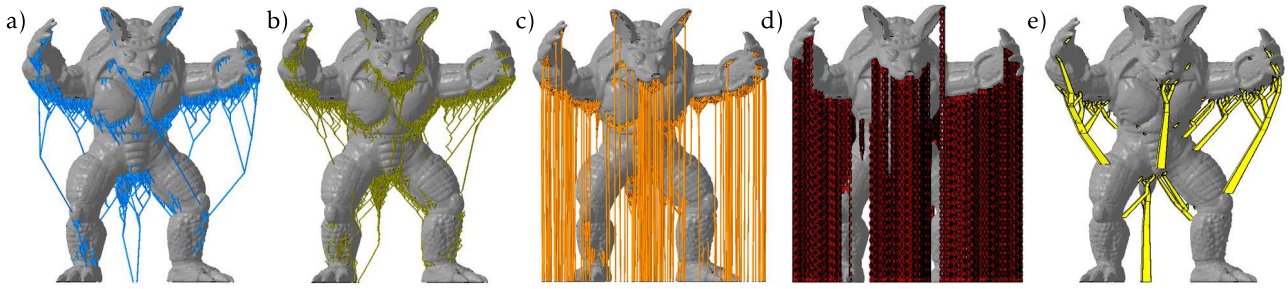

Fig. 16. Armadillo external supports generated by our framework (a), by Meshmixer (b), by SLA strategy (c), by SLM strategy (d), and by Vanek et al. [30] (e).

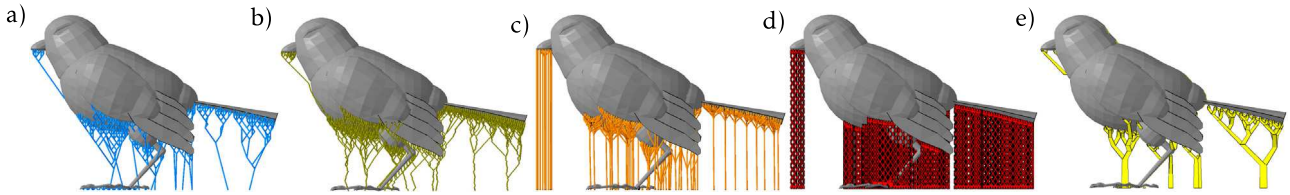

Fig. 17. Bird external supports generated by our framework (a), by Meshmixer (b), by SLA strategy (c), by SLM strategy (d), and by Vanek et al. [30] (e).

Table 5
Volumes comparison between solutions obtained by the proposed framework and the support structures generated by the other strategies.

	Stem supports		Turbine supports		Bunny supports		Armadillo supports	Bird supports
	internal	external	internal	external	internal	external	external	external
Volumes (in mm^3)								
LS^2DO using GA	420	1300	45	3130	210	310	740	810
Schmidt et al. [29], Meshmixer	460	1450	85	3450	245	390	750	1000
SLA strategy	565	1850	70	6990	300	400	940	2280
SLM strategy	630	3100	90	5760	380	470	1370	3650
Vanek et al. [30]	N/A	N/A	N/A	N/A	N/A	N/A	30*	20*
Volume reductions								
LS^2DO wrt Meshmixer	−9%	−10%	−47%	−9%	−14%	−21%	−1%	−23%
LS^2DO wrt SLA strategy	−26%	−30%	−36%	−55%	−30%	−23%	−27%	−181%
LS^2DO wrt SLM strategy	−33%	−58%	−50%	−46%	−45%	−34%	−85%	−351%

*with 10 μm wall thicknesses.

The SLM strategy is mentioned here because it is the main support strategy used with LBM technology. It consist of vertical thin walls, perforated with rhombus shapes. The generated supports are intuitively much denser than the tree-like support structure, because they are also supposed to withstand the part residual stresses. They are also more difficult to remove than the support structures generated by the proposed LS^2DO technique. However, the comparison of the two gives an idea of the optimization range for the support volume: the best trade-off between dense supports to minimize the part deformation, and light supports only sustaining the overhangs, should lie between supports generated by these two strategies.

More precisely, Table 5 compares the volume of the supports obtained by using our framework and the volume of the support structures generated by Meshmixer, by the SLM and the SLA supporting strategies. It clearly shows that the proposed framework enables the generation of the support structures with the lowest volume. Table 5 also indicates the volume of the support

Table 6
Computation times comparison between solutions obtained by the proposed framework and the support structures generated by the other strategies.

	Stem supports	Turbine supports	Bunny supports	Armadillo supports	Bird supports
	Computation times				
LS^{2}DO using GA	75 min	48 min	26 min	33 min	18 min
Schmidt et al. [29], Meshmixer	30 min	154 min	21 min	76 min	83 min
SLA strategy	3 min 21 s	25 s	11 s	9 s	2 s
SLM strategy	1 min 10 s	3 min 5 s	20 s	11 s	4 s
Vanek et al. [30]	N/A	N/A	N/A	161 min	80 min

structures provided by Vanek et al. [30] for the Amarillo and the Bird models only, the code of Vanek et al. not being available (N/A) to test it on the other examples. However, these structures present wall thicknesses of approximatively 10 μm on their whole height, which is technically unmanufacturable in LBM or in FDM at the moment. Therefore, these volumes cannot be fairly compared to the ones of the supports generated by our framework, and are mentioned only on an indicative basis.

5.3.2. Time comparison

A comparison of the computation times required to generate the previously presented support structures is detailed in Table 6. For the Stanford Bunny, the Stem and the Turbine parts, these results include the times to generate both the internal and the external support structures. The results of the LS^{2}DO, the Meshmixer, the SLA and the SLM strategies have been obtained with an Intel®Core™ i7-4710HQ CPU at 2.50 GHz, with 16 GB of RAM, and an Nvidia GeForce GTX 970 M GPU. These characteristics are similar to the ones used by Vanek et al. [30] to obtain their support structures (mentioned in the last line of Table 6). Therefore, the computation times can fairly be compared with respect to hardware characteristics. It is also important to mention that, with the actual implementation of the proposed framework, no GPU computation is realized.

These results demonstrate that the software solutions used by industry to generate support structures are the fastest (i.e. the SLA and SLM strategies). Besides probably using GPU computing, these large gaps can be explained by the fact that their resulting structures are less optimized than the ones proposed by the other methods (as demonstrated by Table 5). However, it can be noticed that the proposed framework is faster than the work of Schmidt et al. [29] implemented in Meshmixer on some test cases, but not on all of them. This might be explained by the fact that their support structures are sometimes going around the part (as detailed in Section 5.3.1): imposing such constraints requires more computation and thus greater optimization times.

In comparison to the work of Vanek et al. [30], the presented framework is faster. This could be due to the difference in the nature of the two considered problems, the one identified in this article being discrete, whereas the one solved by Vanek et al. being continuous. Furthermore, the overhang distance used in their work is greater than the one used here, because the AM technology differs (LBM versus FDM). If they were using the same overhang distance as the one used in this article, the number of connection points on the overhang surfaces would be greater, and their optimization would take even more time to run. The comparison of Table 6 is thus not completely fair, to our own disadvantage.

As a conclusion, besides generating support structures with lower volumes, the proposed framework is also showing reduced computation times in comparison to other state-of-the-art academic works. However, it is not yet competing with GPU-using commercial software. Nevertheless, besides implementing GPU computation, the proposed framework computation time could be optimized by coupling the GA with a heuristic algorithm, or by creating specific GA operators (for the Crossover or the Mutation operator for example), more adapted to the specific LS^{2}DO problem.

Fig. 18. The Stem test case printed in LBM and sawn in two to make internal supports visible.

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, an initial manufacturable lattice structure is generated under the overhanging areas. Then, a GA optimizes this lattice by removing the maximum number of beams, while ensuring that all the areas to support are still sustained. Naturally, working on such tree-like structures also contributes to ease the removal of the external supports during the finishing step.

This article has presented various results. The GA control parameters values the most suited for the LS^{2}DO problem have been selected through a DoE. The internal and external support structures of five test cases have been successfully generated and also manufactured, and their volumes have been compared to the ones of support structures generated by several other state-of-the-art strategies, underlining the interest of the developed algorithm in terms of volume optimization. Other approximation or meta-heuristic algorithms could also be used to find an approximate solution to the DST problem and compared to the GA presented in this article. However, through the compared results, it has been shown that using a GA to to find a solution to the LS^{2}DO problem performs already better than the traditional support generation strategies. The computation times have also been compared, and if commercial solutions perform faster but with less optimized volumes, the proposed approach is faster than other academic methods.

As a perspective for this research, the convergence of the GA could be further improved. For example, a heuristic search could be implemented in order to generate better initial populations, or a quick local search could be done after each crossover and mutation, in order to obtain better child chromosomes.

Furthermore, in order to decrease the computation time of the GA, the various overhanging areas could be optimized by stages: once a first quick optimization is completed, the overhanging areas with no common support structures (two disjoint subgraphs) could be separated in various clusters, and each cluster could be optimized again with a normal optimization. Because each cluster would contain less overhanging areas and less initial beams, their optimization would converge exponentially quicker,

resulting in a reduced overall computation time. Naturally, such a decomposition strategy could also benefit from an ad-hoc GPU implementation.

In order to further improve the optimization of the support structures, the objective function (that only includes the material volume) could be extended, by taking into account the support removal and finishing costs. However, these costs can be hard to estimate because they depend on many factors (e.g. the tools used, the training of the operator).

Finally, because the deformation and thermal accumulation problems have been left aside, the proposed framework is only a first 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 manufactured part.

References

[1] Tofail SA, Koumoulos EP, Bandyopadhyay A, Bose S, O'Donoghue L, Charitidis C. Additive manufacturing: Scientific and technological challenges, market uptake and opportunities. Mater Today 2018;21(1):22–37.

[2] Dilberoglu UM, Gharehpapagh B, Yaman U, Dolen M. The role of additive manufacturing in the era of industry 4.0. Procedia Manuf 2017;11:545–54.

[3] ASTM F2792-12a. Standard terminology for additive manufacturing technologies. 2012, Withdrawn 2015.

[4] Panesar A, Abdi M, Hickman D, Ashcroft I. Strategies for functionally graded lattice structures derived using topology optimisation for additive manufacturing. Addit Manuf 2018;19:81–94.

[5] Brooks H, Brigden K. Design of conformal cooling layers with self-supporting lattices for additively manufactured tooling. Addit Manuf 2016;11:16–22.

[6] Hussein A, Hao L, Yan C, Everson R, Young P. Advanced lattice support structures for metal additive manufacturing. J Mater Process Technol 2013;213(7):1019–26.

[7] Thomas D. The development of design rules for selective laser melting. [Ph.D. thesis], 2009.

[8] Langelaar M. An additive manufacturing filter for topology optimization of print-ready designs. Struct Multidiscip Optim 2017;55(3):871–83.

[9] Calignano F. Design optimization of supports for overhanging structures in aluminum and titanium alloys by selective laser melting. Mater Des 2014;64:203–13.

[10] Järvinen J-P, Matilainen V, Li X, Piili H, Salminen A, Mäkelä I, Nyrhilä O. Characterization of effect of support structures in laser additive manufacturing of stainless steel. Physics Procedia 2014;56:72–81.

[11] Jhabvala J, Boillat E, André C, Glardon R. An innovative method to build support structures with a pulsed laser in the selective laser melting process. Int J Adv Manuf Technol 2012;59(1–4):137–42.

[12] Krol TA, Zaeh MF, Schilp J, Seidel C. Computational-efficient design of support structures and material modeling for metalbased additive manufacturing. In: Ansys conference & 29th CADFEM users meeting. 2011, p. 12.

[13] Krol TA, Zaeh MF, Seidel C. Optimization of supports in metal-based additive manufacturing by means of finite element models. 2012, p. 12.

[14] Jin Y-A, He Y, Fu J-Z. Support generation for additive manufacturing based on sliced data. Int J Adv Manuf Technol 2015;80(9–12):2041–52.

[15] Huang X, Ye C, Mo J, Liu H. Slice data based support generation algorithm for fused deposition modeling. Tsinghua Sci Technol 2009;14:223–8.

[16] Huang P, Wang C, Chen Y. Algorithms for layered manufacturing in image space. In: Advances in computers and information in engineering research, vol. 1. 2014.

[17] Crump SS, Comb JW, Priedeman Jr WR, Zinniel RL. Process of support removal for fused deposition modeling. US Patent; 1996, 5, 503, 785.

[18] Qian B, Lichao Z, Yusheng S, Guocheng L. Support fast generation algorithm based on discrete-marking in rapid prototyping. In: Affective computing and intelligent interaction. 2012, p. 683–95.

[19] Cheng B, Chou K. Geometric consideration of support structures in part overhang fabrications by electron beam additive manufacturing. Comput Aided Des 2015;69:102–11.

[20] Cheng B, Chou YK. Overhang support structure design for electron beam additive manufacturing. ASME; 2017, V002T01A018.

[21] Cooper K, Steele P, Cheng B, Chou K. Contact-free support structures for part overhangs in powder-bed metal additive manufacturing. 2015, p. 12.

[22] Gan M, Wong C. Practical support structures for selective laser melting. J Mater Process Technol 2016;238:474–84.

[23] Boyard N. Méthodologie de conception pour la réalisation de pièces en fabrication additive. [Ph.D. thesis], 2015.

[24] Hussein A. The development of lightweight cellular structures for metal additive manufacturing. [Ph.D. thesis], 2013.

[25] Cloots M, Spierings AB, Wegener K. 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, p. 631–43.

[26] Li D, Dai N, Jiang X, Shen Z, Chen X. Density aware internal supporting structure modeling of 3d printed objects. IEEE; 2015, p. 209–15.

[27] Lee J, Lee K. Block-based inner support structure generation algorithm for 3d printing using fused deposition modeling. Int J Adv Manuf Technol 2016;2151–63.

[28] Swaelens B, Pauwels J, Vancraen W. Method for supporting an object made by means of stereolithography or another rapid prototype production method. US Patent; 1997, 5, 595, 703.

[29] Schmidt R, Umetani N. Branching support structures for 3d printing. In: ACM SIGGRAPH 2014 studio. ACM; 2014, p. 9.

[30] Vanek J, Galicia JAG, Benes B. Clever support: Efficient support structure generation for digital fabrication. Comput Graph Forum 2014;33(5):117–25.

[31] Vaidya R, Anand S. Optimum support structure generation for additive manufacturing using unit cell structures and support removal constraint. Procedia Manuf 2016;5:1043–59.

[32] Dumas J, Hergel J, Lefebvre S. Bridging the gap: Automated steady scaffoldings for 3d printing. ACM Trans Graph 2014;33(4):10.

[33] Shen Z-H, Dai N, Li D-W, Wu C-Y. Bridge support structure generation for 3d printing. In: World scientific. 2016, p. 141–9.

[34] Stava O, Vanek J, Benes B, Carr N, Měch R. Stress relief: Improving structural strength of 3d printable objects. ACM Trans Graph 2012;31(4):48.

[35] Wu J, Wang CC, Zhang X, Westermann R. Self-supporting rhombic infill structures for additive manufacturing. Comput Aided Des 2016;80:32–42.

[36] Lee M, Fang Q, Cho Y, Ryu J, Liu L, Kim D-S. Support-free hollowing for 3d printing via voronoi diagram of ellipses. Comput Aided Des 2018;101:23–36.

[37] Xie Y, Chen X. Support-free interior carving for 3d printing. Visual Inform 2017;1(1):9–15.

[38] Chougrani L, Pernot J-P, Véron P, Abed S. Lattice structure lightweight triangulation for additive manufacturing. Comput Aided Des 2017;90:95–104.

[39] Hauptmann M, Karpiński M. A compendium on steiner tree problems. Inst. für Informatik; 2013.

[40] Watel D. Approximation de l'arborescence de steiner. [Ph.D. thesis], Versailles-St Quentin en Yvelines; 2014.

[41] Halperin E, Krauthgamer R. Polylogarithmic inapproximability. In: Proceedings of the thirty-fifth annual ACM symposium on theory of computing. ACM; 2003, p. 585–94.

[42] Charikar M, Chekuri C, Cheung T, Dai Z, Goel A, Guha S, Li M. Approximation algorithms for directed steiner problems. In: Proceedings of the 9th annual ACM-SIAM symposium on discrete algorithms. 2000, p. 15.

[43] Zelikovsky A. A series of approximation algorithms for the acyclic directed steiner tree problem. Algorithmica 1997;18(1):99–110.

[44] Bravo-Azlan H, Candia-Vejar A. A metaheuristic solution to a constrained steiner tree problem. In: Computer science society, 1997 proceedings., XVII international conference of the chilean. IEEE; 1997, p. 16–20.

[45] Chen Z-H, Hou W-G, Dong Y. The minimum steiner tree problem based on genetic algorithm. In: International conference on modeling, simulation and optimization (MSO 2018). p. 5.

[46] Singh K, Sundar S. Artifical bee colony algorithm using problem-specific neighborhood strategies for the tree t-spanner problem. Appl Soft Comput 2018;62:110–8.

[47] Ge Y, Shan F, Liu Z, Liu W. Optimal structural design of a heat sink with laminar single-phase flow using computational fluid dynamics-based multi-objective genetic algorithm. J Heat Transf 2017;140(2). 022803.

[48] Bhoskar MT, Kulkarni MOK, Kulkarni MNK, Patekar MSL, Kakandikar G, Nandedkar V. Genetic algorithm and its applications to mechanical engineering: A review. Mater Today: Proc 2015;2(4–5):2624–30.

[49] Renner G, Ekart A. Genetic algorithms in computer aided design. Comput Aided Des 2003;(35):18.

[50] Giacomelli D. Geneticsharp. 2018.