Computer-Aided Design 115 (2019) 218–230

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

# Path Planning of a Type of Porous Structures for Additive Manufacturing[☆],[☆☆]

Xiaoya Zhai, Falai Chen [*]

*School of Mathematical Sciences, University of Science and Technology of China, Hefei, Anhui 230026, PR China*

ARTICLE INFO

*Article history:*
Received 1 October 2018
Received in revised form 9 May 2019
Accepted 14 June 2019

*Keywords:*
Porous structures
Path planning
Traveling salesman problem (TSP)
Genetic algorithm

ABSTRACT

Porous structures are widely used in our daily life due to their special properties such as higher rigidity and lightweight. With the popularity of additive manufacturing (AM), producing porous structures by AM shows great advantages over traditional manufacturing technologies. In this paper, we first introduce a framework to model a type of porous structures which contain lots of small holes. Then we propose an efficient path planning algorithm for this type of porous structures. The algorithm consists of the following steps: for each slice of a porous structure, we first subdivide the domain of the slice into subdomains by generalized Voronoi diagram such that each subdomain contains exactly one hole, and then we carry out dual operation to obtain a partition of the domain into a set of subregions. A route is then found to traverse every subregion by solving a traveling salesman problem (TSP) using genetic algorithm (GA), and along the route subregions are merged into larger ones to improve printing efficiency. Finally we fill each merged subregion with a single Fermat spiral curve and further optimize smoothness and uniformity of the filling path. We demonstrated a variety of testing examples and the experimental results showed the superiority and effectiveness of our method in terms of material cost, printing time and structure stability.

## 1. Introduction

Porous structures are common in nature, such as bones, cork, bee hives, corals. Because of their good properties such as lower relative density, higher strength, lightweight, sound insulation and heat insulation, they can improve mechanical properties, such as strength and stiffness, while reducing density. Porous structures are extensively used in aerospace industry, biomedicine, building constructions and so on. Biological scaffolds are typical medical porous structures which can assist patients recuperate their health. Porous structures are excellent buffer structures which can be used for aircraft cushions.

However, manufacturing porous structures is a challenging problem by traditional techniques such as chemical process and subtractive manufacturing which essentially uses a cutting tool to cut materials along a given path. The emerging additive manufacturing (AM) or 3D printing provides a good alternative. Additive manufacturing converts a 3D model into 2D slices and then builds up the whole model layer by layer. Thus it is particularly suitable for complex models which contain lots of structures inside. The key problem for manufacturing porous structures using 3D printing is to design a reasonable printing path for each layer.

Currently, there are several popular path planning methods. Zigzag and contour-parallel path planning methods are the two most popular methods in AM with their own pros and cons respectively. Zigzag paths (Fig. 1(a)) are simple to implement, but they are often discontinuous and have serrated boundaries which affect printing accuracy. Contour-parallel paths (Fig. 1(b)) can effectively guarantee geometric accuracy of boundaries, but they often create lots of small fragments. Maze patterns (Fig. 1(c)) can also be used as filling paths but it is a complicated task to design a maze pattern for an arbitrary boundary. Space filling paths such as Hilbert filling paths (Fig. 1(d)) and spiral filling paths (Fig. 1(e) and Fig. 1(f)) can achieve global continuity but have low computational efficiency. Despite many efforts that have been contributed for path generation, for complex geometries such as porous structures, it is still a difficult task to design a high-quality printing path. The goal of the current paper is to propose an effective path-planning algorithm for porous structures.

The main contributions of this paper are as follows:

- An implicit representation is proposed to model a type of porous structures which contain lots of small holes.

[☆] This paper has been recommended for acceptance by Charlie Wang.
[☆☆] No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.cad.2019.06.002.
[*] Corresponding author.

*E-mail addresses:* xiaoya93@mail.ustc.edu.cn (X. Zhai), chenfl@ustc.edu.cn (F. Chen).

https://doi.org/10.1016/j.cad.2019.06.002
0010-4485/© 2019 Elsevier Ltd. All rights reserved.

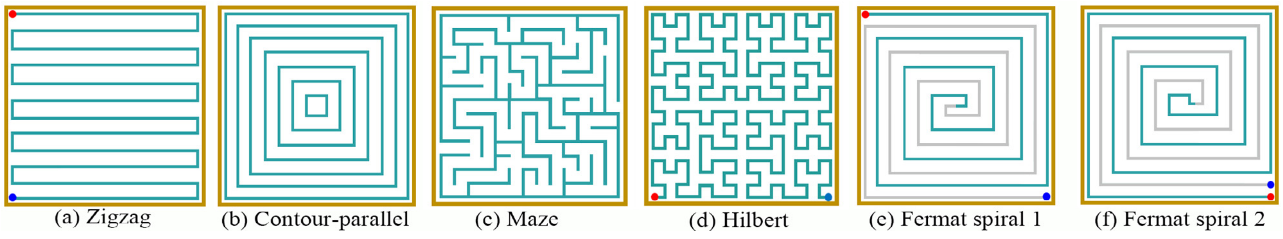

**Fig. 1.** Various filling paths.

- A novel path generation algorithm is developed for such type of porous structures by solving a traveling salesman problem.
- A variety of testing examples are presented to demonstrate the superiority and effectiveness of our method in terms of material cost, printing time and structure stability.

The structure of this paper is organized as follows. In Section 2, we review some related work about path planning and porous structures. In Section 3, we propose an implicit representation for a type of porous structures which contain lots of small holes. In Section 4, we describe a new path generation algorithm for the porous structures described above. In Section 5, experimental results are demonstrated and comparisons are made between various methods on several factors such as algorithm efficiency, time and material cost, path continuity, sharp turns, fill rates, physical property and visual effects, where material cost and printing time are the most important factors. We conclude the paper with a summary and some future work in Section 6.

## 2. Related work

### 2.1. Path planning

Path planning is an essential step in additive manufacturing. A path planning method should seek good performance in the following aspects. Firstly, the path needs to be globally continuous. Printing material will be pumped back in discontinuous sections if path is not globally continuous. The nozzle of the printer could suffer from the process of lifting, rapid positioning and falling, which increases the printing time and affects the overall printing quality. Secondly, sharp turns should be avoided as much as possible. Sharp turns would slow down the nozzle causing time wastage and create tiniest interspace. Thirdly, the path should be as uniform as possible. Over-fills increase material cost and under-fills affect printing quality. Other factors include time and material consumption, etc. These factors should be as small as possible under a certain filling rate. In the following, we review some related works on path planning.

**Zigzag paths** It is the most typical path planning method used in AM. Zigzag paths consist of line segments which are parallel to one direction. This method has been applied in commercial printers because of high computational efficiency. The research mainly focuses on how to determine the optimal inclination degree. Park et al. [1] found the optimal inclination by considering the shape of the printing area as well as the path interval. Rajan et al. [2] provided a method for efficiently computing an optimal inclination degree. They minimized the number of scan line segments when the region was bounded by straight line segments and/or circular arcs. Kim [3] applied an intersection point graph to generate a path, and then modified it to maintain a constant material removal rate to achieve a constant cutting force and avoid chatter vibration. Jin et al. [4] proposed a direction-parallel path planning method with full consideration of specific technical characteristics in fused deposition modeling (FDM). Zigzag paths have high computation efficiency, but they produce many sharp turns and staircase effects which influence the printing quality.

**Contour-parallel paths** Contour-parallel paths are comprised of a series of contours which are offsets to the input boundaries. Such paths not only possess geometric precision but also avoid the staircase effects. Farouki et al. [5] constructed equidistant offset curves based on rational representations of model boundaries. It significantly enhances the accuracy and speed of offset computations. Yang et al. [6] introduced an efficient equidistant path generation method which consists of three steps: domain partition, offset generation and intersection processing. This method can bring significant improvements to printing process both in processing efficiency and printing quality. Abdullah et al. [7] proposed a contour-parallel path generation method based on an ant colony optimization technique to generate a clear tool path that removes the entire uncut region in contour parallel machining at minimum cutting time. One major disadvantage of contour-parallel methods is that they may produce lots of small uncut regions and inflection points which create under-fills and increases printing time.

**Space-filling paths** There are many kinds of space-filling paths, such as Pinao curves, Hilbert curves and spiral curves. S.H. Nair et al. [8] presented a systematic strategy to produce uniform or non-uniform Hilbert's space-filling curves in the space with obstacles (or holes). The Hilbert curve is not suitable for additive manufacturing due to too many inflection points. C. Fleming et al. [9] designed a post-processing greedy algorithm to reorder path instructions in order to reduce the distance traveled between subsequent space-filling curves and layers. Held et al. [10] generated a spiral curve for a domain without islands by interpolating growing disks placed on the medial axis of the boundary. They further extended their method to multiply-connected domains [11] by decomposing an arbitrarily complex domain into simpler ones and filling each subregion with one distorted spiral curve. However, such paths cannot be used for additive manufacturing directly due to uneven intervals. Zhao et al. [12] made further improvements over the above algorithm. They decomposed a domain into a set of subregions according to positions of offset curves. Each subregion was filled with a single continuous Fermat spiral curve by rerouting and connecting, and finally a global continuous path could be obtained. However, the method requires that start and exit points should locate at approximately the same location, and rate of under-fills by the method could be high. In summary, space-filling paths are effective to avoid self-intersection and make the whole path continuous. However, it may take a long time to generate filling paths for complex geometries.

**Hybrid-filling paths** In some situations, a single method cannot produce satisfactory filling paths, but the combination of different algorithms can achieve better results. Jin et al. [13] proposed a method that generates contour-parallel paths to improve the geometrical quality and zigzag paths for the internal area of the model to simplify computation and fabrication processes. Ding et al. [14] decomposed 2D geometries into a set of convex polygons, and for each convex polygon, an optimal inclination is determined and a continuous path is generated by using a combination of zigzag and contour-parallel paths. The final path is formed by connecting all the sub-paths. Ozbolat et al. [15] introduced an algorithm that generates a bilayer pattern of zigzag and spiral paths for porous structures with internal features. This is a practical method to print functionally graded materials.

### 2.2. Porous structures

Porous structures have directional or random holes like foam, lotus, honeycomb. They have the characteristics of small proportion, rigidity, vibration absorption and so on. There have been several attempts on how to generate porous models. Lu et al. [16] introduced a honeycomb-cells structure by a hollowing optimization algorithm to reduce the material cost and weight of a given object. Wu et al. [17] presented a method to generate bone-like porous structures based on structural optimization to obtain maximum stiffness and lightweight. Ying et al. [18] proposed an algorithm for modeling anisotropic porous structures based on anisotropic centroidal Voronoi tessellations which have better adaption with the stress tensor field.

## 3. Porous structures represented by implicit functions

In Computer Aided Design, NURBS is a standard tool to represent geometric objects. However, it is a very difficult task for NURBSs to model complicated geometry and topology such as porous structures. Polygonal models are another common representation in geometric modeling. Yet it would take a huge number of faces to model a porous structure with acceptable accuracy. Furthermore, parametric and polygonal representations have disadvantages in 3D printing applications. To convert a 3D parametric or polygonal model into 2D slices, geometrical and topological errors often occur, which would result in incorrect printing or even failure. In contrast, implicit representation is well suited to model complicated geometry with arbitrary topology and inner structures [19], and it is more efficient and robust to convert a 3D implicit representation into 2D slices [20]. In this section, we propose an implicit representation to model a type of porous structures which contain lots of small holes inside.

Firstly, we represent the boundary of a porous structure by a trivariate spline function of tri-degree $(d_1, d_2, d_3)$:

$$f_b(x, y, z) = \sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{k=0}^{l} c_{ijk}N_i(x)N_j(y)N_k(z) = 0, \tag{1}$$

where $\{c_{ijk}\}_{i,j,k=0}^{m,n,l}$ are the control coefficients, $N_i(x), N_j(y), N_k(z)$ are the B-spline basis functions along $x$, $y$, and $z$ directions of degrees $d_1, d_2, d_3$, respectively. In practice, we usually choose $d_1 = d_2 = d_3 = 3$.

Next, we represent the inner holes using blobby models [21]. Let $\boldsymbol{P} = \{p_1, p_2, \ldots, p_N\}$ be a set of seed points in the interior of a model $M$, where $p_i = (x_i, y_i, z_i)$, $i = 1, 2, \ldots, N$. Define a potential function for each point $p_i$:

$$f_i(x, y, z) = \begin{cases} 1 - \dfrac{3d_i^2}{r_i^2}, & 0 \le d_i < \dfrac{r_i}{3}, \\ \dfrac{3}{2}(1 - \dfrac{d_i}{r_i})^2, & \dfrac{r_i}{3} \le d_i < r_i, \\ 0, & r_i \ge d_i \end{cases} \tag{2}$$

where

$$d_i^2 = \|q - p_i\|^2 = \frac{(x - x_i)^2}{a_i^2} + \frac{(y - y_i)^2}{b_i^2} + \frac{(z - z_i)^2}{c_i^2},$$

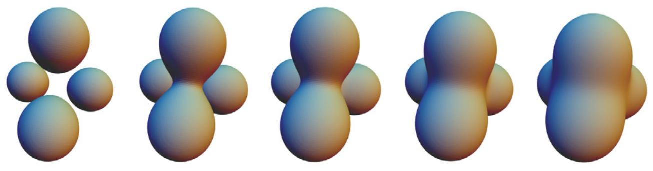

**Fig. 2.** The iso-surfaces for different values of $\kappa$.

$r_i > 0$ is the radius with respect to the point $p_i$, $d_i$ is the scaled distance from one point $q$ to the point $p_i$, and $a_i, b_i, c_i > 0$ are scaling factors along three coordinate directions. When $a_i = b_i = c_i = 1$, $d_i$ is the usual Euclidean distance. Now the blobby model generated by the point set $\boldsymbol{P}$ is defined as

$$f_{in}(x, y, z) := \sum_{i=1}^{N} f_i(x, y, z) - \kappa = 0, \tag{3}$$

where $\kappa$ is a constant. For different choices of $\kappa$, one obtains different blobby models. Fig. 2 illustrates the iso-surfaces with four seed points for five different values of $\kappa$.

Now the porous structure is defined by the following implicit function

$$f(x) = max(f_b, -f_{in}) \tag{4}$$

which represents the Boolean operation of two implicit geometries defined by $f_b = 0$ and $f_{in} = 0$.

For a given model $M$, one can convert the representation of the boundary surface of $M$ into trivariate splines using techniques in [22]. The inner porous structures are determined by the locations and radii of the seed points. For our demonstration purpose, we randomly generate the seed points $\boldsymbol{P}$ by Lloyd′s relaxation method [23]. To determine the radius $r_i$ of the point $p_i$, we compute the Voronoi diagram of the seed points and calculate the shortest distance $h_i$ from $p_i$ to the inner wall of the Voronoi cell(a polyhedron), and set $r_i = e_i * h_i$, where $e_i$ is a random number in $(0, 1)$. $a_i, b_i, c_i$ are also random numbers between 0.5 and 1.5 to generate ellipsoid shapes. The inner structure is thus defined by (3) (Fig. 3(e)).

**Remark 3.1.** In this section, we propose a general framework to model a type of structures using implicit representations. The inner structure is determined by the locations and radii of the seeds. We randomly generate the locations and radii of the seeds solely for demonstrating the path-planning algorithm presented in the next section. In practice, the locations and radii of the seeds can be obtained by shape and topological optimization technique, which is a widely investigated and on-going research area [24]. How to optimize the seeds is a very important research topic and is far beyond the scope of the current work.

## 4. Path generation

In this section, a new path planning algorithm is proposed for printing porous models constructed in Section 3. As a first step, we need to generate a series of 2D slices of a porous structure. For implicit representation, this can be done efficiently by utilizing the incremental method [20,25]. Fig. 4 shows partial results of slicing a porous structure. It should be emphasized that the algorithm presented in this section works for any representations (parametric, polygonal and implicit) of the geometric model.

Now suppose a 2D slice (a domain) is generated. Our method consists of the following steps:

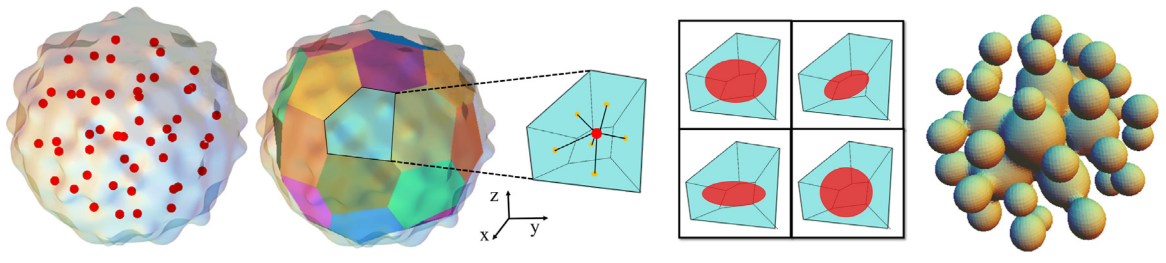

**Fig. 3.** The process of generating implicit representation of a porous structure. (a) Seed points distributed in a model. (b) Voronoi cells of seed points. (c) One cell. (d) A blobby in a cell. (e) Inner holes represented by blobby models.

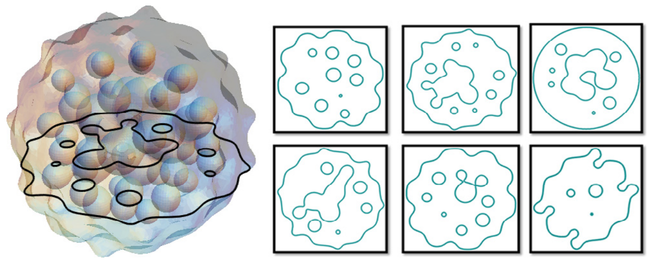

**Fig. 4.** Cross sections of a porous model.

1. Divide the domain into a set of subregions by generalized Voronoi diagram [26] and dual operation;
2. Find a route to connect all the subregions by solving a traveling salesman problem (TSP);
3. Merge subregions by maximum approximate ellipse principle.
4. Fill each subregion with an optimal Fermat spiral curve and optimize smoothness and uniformity of the path.

The overall process of our algorithm is shown in Fig. 5. In the following, we will describe each step in detail.

### 4.1. Regional segmentation

Regional segmentation decomposes a topologically complex domain into simpler ones. Let $D$ be the domain of the given 2D slice, and $H_1, H_2, \dots, H_r$ be the inner holes inside $D$. We first subdivide the domain $D$ into $r$ subdomains $D_1, D_2, \dots, D_r$ such that each subdomain $D_i$ encloses one hole $H_i$. Define

$$D_i := \{p \in D \mid d(p, \partial H_i) \le d(p, \partial H_j),\ j = 1, \dots, r,\ j \ne i\} \tag{5}$$

that is, $D_i$ consists of those points whose distances to the boundary of $H_i$ are less than or equal to the distances to the boundaries of other holes. $\{D_i\}_{i=1}^r$ form a partition of the domain $D$, and we call it Generalized Voronoi Diagram (GVD), and $D_i$ a cell of the GVD.

To compute the GVD of $D$, we sample $m$ points $q_j$ ($j = 1, 2, \dots, m$) inside $D$ and compute the distance $d(q_j, \partial H_i)$ of $q_j$ to the boundary of the hole $H_i$. If $d(q_j, \partial H_i) \le d(q_j, \partial H_{i'})$ for $i' = 1, 2, \dots, r$, $i' \ne i$, then $q_j \in D_i$. Hence the problem converts into distance computation of a point to the boundary of a hole which is represented by an implicit function.

Let $q_j = (x_j, y_j)$ and assume the boundary of the hole $H_i$ is represented by $g_i(x, y) = 0$. Then the foot point $(x, y)$ of $q_j$ on $\partial H_i$ can be computed by solving

$$g_i(x, y) = 0, \quad g_{ix}(x, y)(y - y_j) - g_{iy}(x, y)(x - x_j) = 0.$$

Newton–Raphson's method can applied to solve the equations efficiently. After the foot point is calculated, the distance $d(q_j, \partial H_i)$ is thus obtained.

To speedup computation, we use bounding box technique to eliminate many distance calculation. Let $B_i$ be the bounding box (or circle) of the inner hole $H_i$, $i = 1, 2, \dots, r$, and denote $L_{ij}^1$ and $L_{ij}^2$ to be the shortest and longest distances between $q_j$ and $\partial B_i$ respectively. Assume that $L_{i_0 j}^2 = \min_i L_{ij}^2$. Then for any $i$ such that $L_{i_0 j}^2 < L_{ij}^1$, we have $d(q_j, \partial H_{i_0}) < d(q_j, \partial H_i)$, and hence we do not have to compute $d(q_j, \partial H_i)$ for such $i$. The details of computing GVD are shown in Algorithm 1.

**Algorithm 1** Algorithm for computing generalized Voronoi diagram

```
Input: A 2D domain $D$ which contains the inner holes $\{H_i\}_{i=1}^r$.
Output: Generalized Voronoi diagram of the domain $D$.
1: Uniformly sample $m$ points $q_j$ ($j = 1, 2, \dots, m$) inside $D$.
2: for $i = 1$ to $r$ do
3:    Compute the bounding box(circle) $B_i$ of the inner hole $H_i$.
4: end for
5: for $j = 1$ to $m$ do
6:    for $i = 1$ to $r$ do
7:       Calculate the shortest and longest distances $L_{ij}^1$ and $L_{ij}^2$ between $q_j$ and $\partial B_i$.
8:    end for
9:    Let $L_{i_0 j}^2 = \min_i L_{ij}^2$, and set $d_j = d(q_j, \partial H_1)$ and $i' = 1$.
10:   for $i = 2$ to $r$ do
11:      if $L_{ij}^1 \le L_{i_0 j}^2$ then
12:         Calculate the distance $d(q_j, \partial H_i)$.
13:         if $d_j > d(q_j, \partial H_i)$ then
14:            Set $q_j = d(q_j, \partial H_i)$ and $i' = i$.
15:         end if
16:      end if
17:   end for
18:   Set $q_j \in D_{i'}$.
19: end for
```

Fig. 6(b) shows the GVD of a domain $D$. From the GVD, it is easy to derive the adjacency information of the holes in $D$. For each pair of adjacent holes, we find a shortest line to connect the two holes. The domain $D$ is thus subdivided into a set of subregions $\{R_i\}_{i=1}^k$ whose boundaries are formed by alternatively connecting the line segments and parts of the hole boundaries. Fig. 6(c) shows the decomposition result of the domain $D$. Notice that each subregion is now a simply connected domain.

### 4.2. Regional traversal

Let us now discuss how to traverse the subregions $\{R_i\}_{i=1}^k$ of a domain $D$ such that a continuous path can be found to fill each subregion sequentially. We first transfer the decomposition of $D = \cup_{i=1}^k R_i$ into a directed graph $G = (V, E)$, where each subregion $R_i$ is represented by a vertex $v_i \in V$, and for each pair

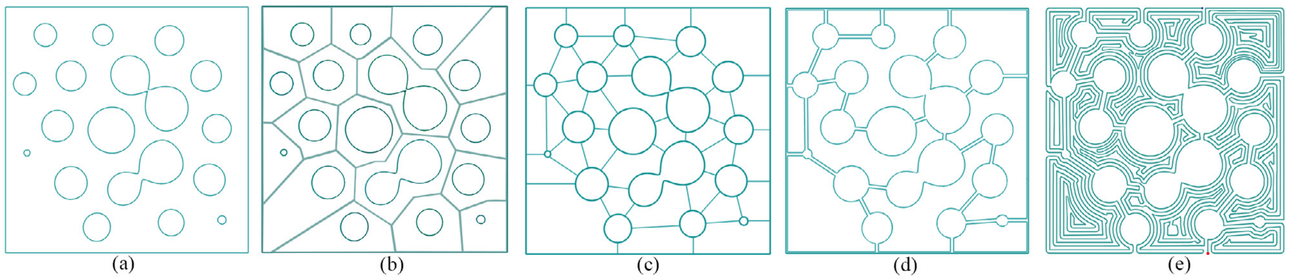

**Fig. 5.** Procedure for the path planning of porous structures. (a) Input 2D slice. (b) Decompose the 2D slice into subdomains. (c) Carry out dual segmentation of (b). (d) Traverse and merge subregions. (e) Fill and optimize subregions with Fermat spirals.

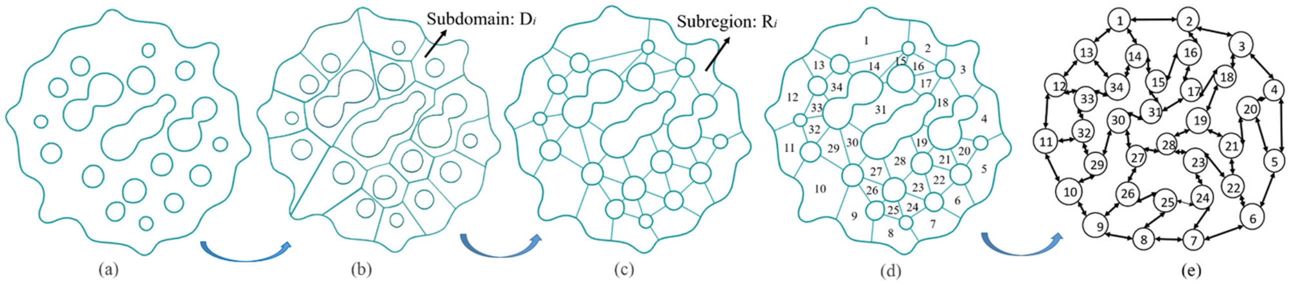

**Fig. 6.** Regional segmentation and traversal. (a) Input domain $D$. (b) GVD of $D$. (c) $D$ is decomposed into subregions. (d) Number each subregion. (e) A graph representation of the decomposition.

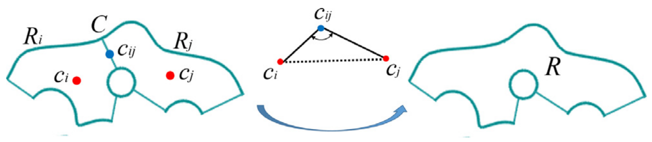

**Fig. 7.** Definition of angle between two subregions and merging process.

of adjacent subregions $R_i$ and $R_j$, it corresponds to two directional edges $e_{ij}, e_{ji} \in E$. Fig. 6(e) shows the representative graph of the domain $D$. Now the problem reduces to find a route in the graph $G$ that traverses all the vertices exactly once, and this is the so-called Hamilton path of $G$. Theoretically, any Hamilton path of $G$ gives a solution to our problem. However, the filling path by such solution is composed of lots of small spiral curves, which influences the printing efficiency. Our idea is to merge small subregions into larger ones as much as possible, and then print each larger subregion sequentially. Specifically, without loss of generality, let $v_1 v_2 \ldots v_k$ be a Hamilton path of $G$ and $R_1 R_2 \ldots R_k$ be the corresponding subregions, we merge consecutive subregions $R_{i_j} \ldots R_{i_{j+1}-1}$ into a larger subregion $R'_j$, $j = 1, 2, \ldots, k'$, where $i_1 = 1$ and $i_{k'+1} = k+1$. Thus the domain $D$ is composed of larger subregions: $D = \cup_{j=1}^{k'} R'_j$. Fill each subregion $R'_j$ with a Fermat spiral curve, we can arrive at a final printing path.

Now the problem reduces to find a Hamilton path in $G$ such that the merging process can produce larger subregions $R'_j$ as much as possible. For that purpose, we first give a definition.

**Definition 4.1.** Let $R_i$ and $R_j$ be two neighboring subregions with $c_i$ and $c_j$ being their centers respectively. Let $C$ be the common boundary of $R_i$ and $R_j$ and $c_{ij}$ be the middle point of $C$. Then we define the angle between $R_i$ and $R_j$ to be $\angle c_i c_{ij} c_j$. See Fig. 7 for a reference.

Two subregions are likely to be merged if the angle between the two subregions is not too far from $\pi$. Now we associate each edge $e_{ij} \in E$ of the graph $G$ with a weight $w_{ij} = 1/(\sin(\theta_{ij}/2))^4$, where $\theta_{ij}$ $(0 < \theta_{ij} \leq \pi)$ is the angle between the two subregions $R_i$ and $R_j$. Thus $G$ becomes a weighted directed graph, and obviously $w_{ij} = w_{ji}$. Our goal is to find a Hamilton path in $G$ with minimal total weights. This leads to the following optimization problem:

$$argmin \quad f(x) = \sum_{(i,j)\in\Gamma} w_{ij} x_{ij} \tag{6}$$

$$\text{s.t.} \sum_{j\in N(i)} x_{ij} = 1, \quad i = 1, 2, \ldots, k, \ i \neq k_1 \tag{7}$$

$$\sum_{i\in N(j)} x_{ij} = 1, \quad j = 1, 2, \ldots, k, \ j \neq k_0 \tag{8}$$

where $\Gamma = \{(i,j) \mid e_{ij} \in E\}$, $N(i) = \{j \mid (i,j) \in \Gamma\}$ is the index set of vertices in the neighborhood of vertex $v_i$, and

$$x_{ij} = \begin{cases} 1, & e_{ij} \text{ is on a Hamilton path} \\ 0, & e_{ij} \text{ is not on a Hamilton path} \end{cases}$$

$k_0$ is the index of starting vertex and $k_1$ is the index of ending vertex. The objective function $f(x)$ is the sum of the weights on a Hamilton path, and the constraints (7) and (8) ensure that at each vertex in the path (except the start and end vertices) there is only one incident edge and one out edge. In addition, there is a limitation that sub-loop cannot be present in the final path.

The optimization problem is the same as traveling salesman problem (TSP), and it is NP-hard to solve. In this paper, we adopt genetic algorithm (GA) to solve this problem. Genetic Algorithm mimics the process of natural selection in the biological world to generate high-quality solutions to optimization problems. In a genetic algorithm, a population of candidate solutions (individuals) to an optimization problem is evolved toward better solutions through three operations on the chromosomes of individuals, namely, selection $O_s$, crossover $O_c$ and mutation $O_m$. Applying the genetic algorithm in our problem, we start with a randomly constructed set of paths $X^0$ that connect all the vertices in $G$. A new set of paths $X^1$ is then generated through the three operations on the set of paths $X^0$–selection, crossover and mutation. Iteratively improve the set of paths $X^t$ by evaluating the fitness function to make the total weights become smaller until no more improvements are possible. In the following, we explain the three operations in detail.

**Parameter selection** The performance of GA is affected by several factors, such as the size of the initial population $M_0$, the selection's strategy $O_s$, the mutation operator $O_m$, the crossover operator $O_c$ and the coding scheme. In this paper, we make the following selection: population size $M_0 = 100$, the rate of selection $Q_r = 0.25$, the rate of mutation $Q_m = 0.2$, the rate of crossover $Q_c = 0.35$ and maximum evolutionary number $N = 500$, and integer chromosomes encoding is adopted.

**Fitness function** Fitness function is a measure to determine if an individual will be inherited into next generation or not, and it has great influence on GA. Here we take the reciprocal of the objective function (9) as the fitness function:

$$F(x^t) = (f(x^t))^{-1}, \tag{9}$$

where $x^t$ is an individual at the $t$th iteration. At each iteration, we try to maximize the fitness of each individual $x^t$.

**Selection operation** Optimal preservation strategy is adopted here. We first calculate the fitness values of all the individuals in the current population $X^t = \{x_1^t, x_2^t, \ldots, x_{M_t}^t\}$, where $x_i^t$ is an individual at time $t$, and $M_t$ is the number of individuals in $X^t$. The individual with the highest fitness does not participate in the crossover operation and mutation operation in later stages, and it will be inherited into next generation directly. Other individuals will be inherited into next generation according to certain probability:

$$P(x_l^t) := \frac{F(x_l^t)}{\sum_{i=1}^{M_t} F(x_i^t)}, \qquad l = 1, 2, \ldots, M_t. \tag{10}$$

If $P(x_l^t) > Q_r$, then the individual $x_l^t$ will be inherited; otherwise, it will be not. Optimum maintaining genetic algorithm can improve the global convergence of GA. Rudolph [27] proved theoretically that the standard genetic algorithm with optimum preservation is globally convergent.

**Crossover operation** This operation aims to simulate the exchange of genes which are particular positions or locus in a chromosome when organisms reproduce. Crossover is the main phase of GA that generates new individuals and it determines the global search ability of GA. The main process of crossover operation is as follows. We first randomly select two parental individuals $x_1$ and $x_2$ (or chromosomes—a Hamilton path in our problem) in the current generation, and generate a random number $p_c$ in the interval [0, 1] with uniform probability. If $p_c > Q_c$, no crossover occurs. Otherwise, the crossover operator is applied. In this paper, we choose two-point crossover operator. Two crossover points with a fixed length between the two points are selected in each parental chromosome ($x_1$ or $x_2$). The gene between two the crossover points is defined as the gene fragment—a part of a path in our problem. We establish a one-to-one correspondence between the two gene fragments of the two parental chromosomes $x_1$ and $x_2$. The child chromosomes are constructed by interchanging the gene fragments of $x_1$ and $x_2$, and replacing the genes outside of the gene fragments one by one according to the corresponding relations on the gene fragments.

**Mutation operation** This operator acts on an individual chromosome and randomly flips one or several genes of the chromosome. It not only determines the local search capability of GA but also effectively prevents local convergence. The procedure is as follows. A random number $p_m$ in the interval [0, 1] is generated with uniform probability. If $p_m > Q_m$, no mutation is applied. Otherwise, we randomly select two genes on the chromosome and swap them.

These three operations are carried out repeatedly until the difference of the selection probabilities between two adjacent generations is less than threshold $\tau$ or the number of iterations reaches a specific number $N$. The outline of GA is shown in Algorithm 2. An example is illustrated in Fig. 6(d), where the traversal sequence obtained by the above algorithm is: $3 \to 18 \to 19 \to 28 \to 27 \to 26 \to 25 \to 24 \to 23 \to 22 \to 21 \to 20 \to 4 \to 5 \to 6 \to 7 \to 8 \to 9 \to 10 \to 11 \to 12 \to 33 \to 32 \to 29 \to 30 \to 31 \to 17 \to 16 \to 15 \to 14 \to 34 \to 13 \to 1 \to 2$.

**Remark 4.1.** In some situation, the graph $G$ may do not have a Hamilton path. In this case, we find a path that contains as many number of vertices as possible. Then subregions whose corresponding vertices are not in the path are merged into subregions whose corresponding vertices are in the path.

**Algorithm 2** Optimal path construction algorithm

```
Input: A graph $G$.
Output: An optimal Hamilton path of $G$.
 1: Set the initial values of the parameters: $M_0 = 100$, $Q_s = 0.25$,
    $Q_c = 0.35$, $Q_m = 0.2$, $Q_r = 0.05$, $N = 500$, $\tau = 0.000005$ and
    $t = 0$.
 2: Randomly generate $M_0$ individuals as the initial population
    $X^0 = \{x_1^0, x_2^0, \cdots, x_{M_0}^0\}$. Calculate the fitness function values
    $F(x_i^0)$ for each individual $x_i^0$ in $X_0$, $0 \le i \le M_0$.
 3: while $\{(t \le N)$ and $(P(X^{t+1}) - P(X^t) > \tau)\}$ or $(t = 0)$ do
 4:   Perform a selection operation $O_s$.
 5:   for $l = 1$ to $N$ do
 6:     if $P(x_l^t) > Q_r$ then
 7:       Pass the individual $x_l^t$ to the next generation.
 8:     end if
 9:   end for
10:   Perform the crossover operation $O_c$ and mutation operation
      $O_m$.
11:   Update $X^{t+1}$ and set $t = t + 1$.
12: end while
```

### 4.3. Regional consolidation

Small print areas tend to have a high proportion of sharp turns and uneven filling. How to merge small subregions into larger print-friendly regions will be taken into account in this subsection. By the regional traversal algorithm presented in the last subsection, we have obtained a sequence of subregions $R_1, R_2, \ldots, R_k$ which comprise the whole domain $D$. Our goal is to merge them into a new sequence of larger print-friendly regions $R'_1, R'_2, \ldots, R'_{k'}$. We define a print-friendly region by the following characteristics: (1) the region is as close to an ellipse as possible, and the ratio of the lengths of the long axis and short axis is bounded above by a constant; (2) the region is a simply connected domain; (3) the boundary of the region is as smooth as possible. For a given region $R$, we first find an ellipse $E$ such that $R$ and $E$ are as close as possible. Let

$$E(x, y) := e_1x^2 + e_2xy + e_3y^2 + e_4x + e_5y + e_6 = 0 \tag{11}$$

be the implicit equation of the ellipse $E$, where $4e_1e_3 - e_2^2 = 1$. We sample a set of points $Q = \{q_1, q_2, \ldots, q_m\}$ on the boundary of $R$ with $q_i = (x_i, y_i)$ and minimize the sum of the squared algebraic distances of the point set $Q$ to $R$:

$$\min \quad \sum_{i=1}^{m} E(x_i, y_i)^2 \tag{12}$$

$$s.t. \quad 4e_1e_3 - e_2^2 = 1$$

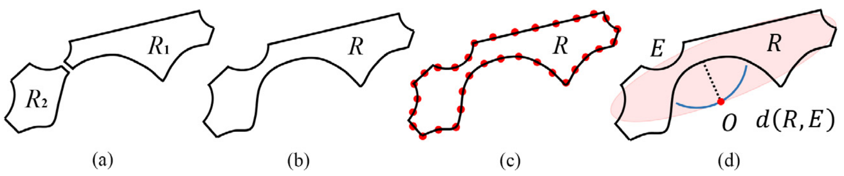

**Fig. 8.** Regional consolidation. (a) Two neighboring subregions $R_1$ and $R_2$. (b) Union of the two subregions $R = R_1 \cup R_2$. (c) Sampling points on $R$. (d) Elliptic fitting and merging.

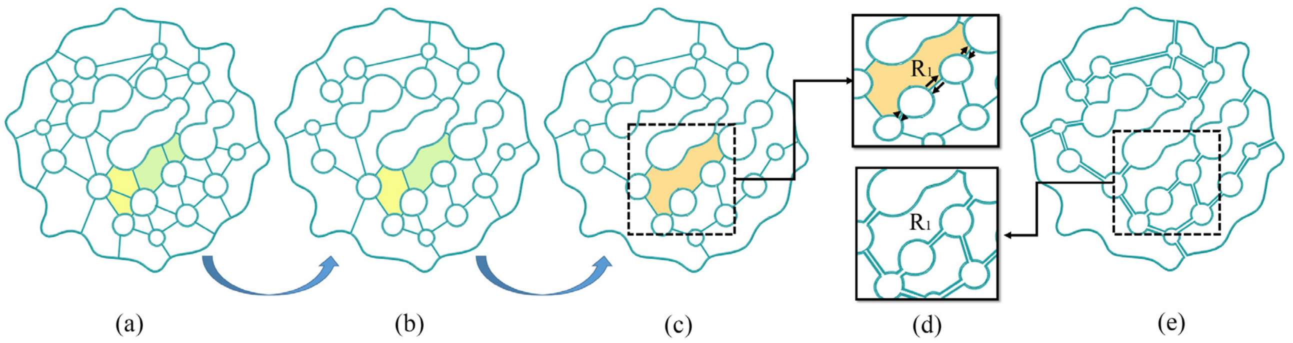

**Fig. 9.** Regional consolidation for an example. (a) Original segmentation; (b) and (c) regional consolidation; (d) close-up views; (e) final result.

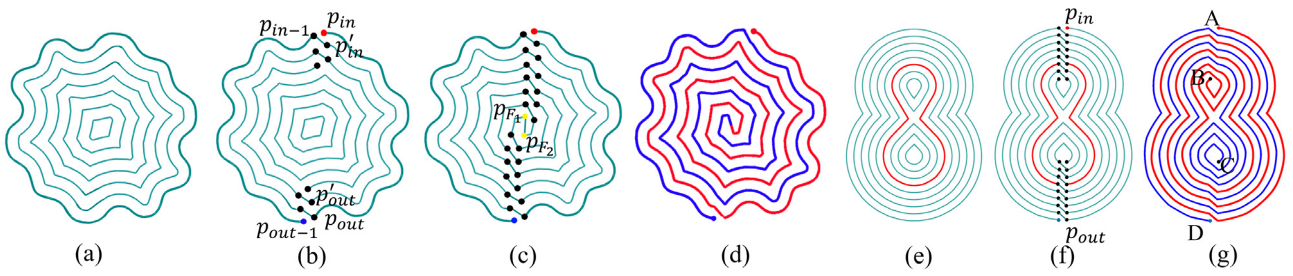

**Fig. 10.** Filling a region with a single local minimum. (a) Original region; (b) Break and reroute the connection relationship; (c) Connect final junction points; (d) The final result. (e)–(g) show that the method does not apply for regions with multiple local minima.

This is simply a quadratic programming problem with 6 variables $e_1, \ldots, e_6$ and it can be easily solved.

After obtaining the ellipse $R$, we use one-sided Hausdorff distance to measure the difference of a region $R$ and the ellipse $E$:

$$d(R, E) := \max_{x \in \partial R} \min_{y \in \partial E} d(x, y).$$

Two neighboring subregions $R_1$ and $R_2$ are merged into a subregion $R$ if

$$d(R, E) < b_E \quad \text{and} \quad a_E < 3b_E \tag{13}$$

where $a_E$ and $b_E$ are the lengths of the long half axis and the short half axis of the ellipse $E$. Fig. 8 shows the process of regional consolidation, while Fig. 9 illustrates the result of regional consolidation for an example.

**Remark 4.2.** Before filling each subregion, the common boundary of two neighboring subregions $R'_i$ and $R'_{i+1}$ should be offset by half of the printing width (as shown in Fig. 9(d)) to avoid overfills in the boundary.

### 4.4. *Fill one subregion*

By previous step, we have obtained a sequence of merged subregions $R'_1, R'_2, \ldots, R'_{k'}$. The problem now is to design a continuous space filling path for a given simple region with arbitrary entry and exit points. Fermat spirals are a good choice for this task. Since the path planning is independent for each subregion, the computation can be done in parallel. In the following, we solve the problem by considering two situations, a region with a single local minimum and multiple local minima. A region is said to have a single local minimum if inside the region all the offset curves of the boundary are simply connected close curves of one loop. Otherwise, it is said to have multiple local minima.

#### 4.4.1. *Fill a region with a single local minimum*

Given a region $R$ with a single local minimum, the entry point $p_{in}$ and exit point $p_{out}$ on the boundary of the region, we are going to construct a continuous Fermat spiral to fill the region $R$. Here are the main steps:

- Compute a series of offset curves $C_1, C_2, \ldots, C_l$ of the regional boundary $\partial R$ with equidistance $d$ by Clipper's algorithm [28].
- Select a point $p_{in-1}$ near $p_{in}$ on the boundary such that the distance between the two points is $d$.
- Select a point $p'_{in}$ on $C_1$ such that $|p_{in}p'_{in}| = d$.
- Connect the two points $p_{in-1}$ and $p'_{in}$, and delete the line segment $p_{in}p_{in-1}$.
- Conduct the same operations for the exit point $p_{out}$, and for all the other offset curves consecutively. Then we finally obtain a continuous path filling the region $R$.

Fig. 10 illustrates the above procedure.

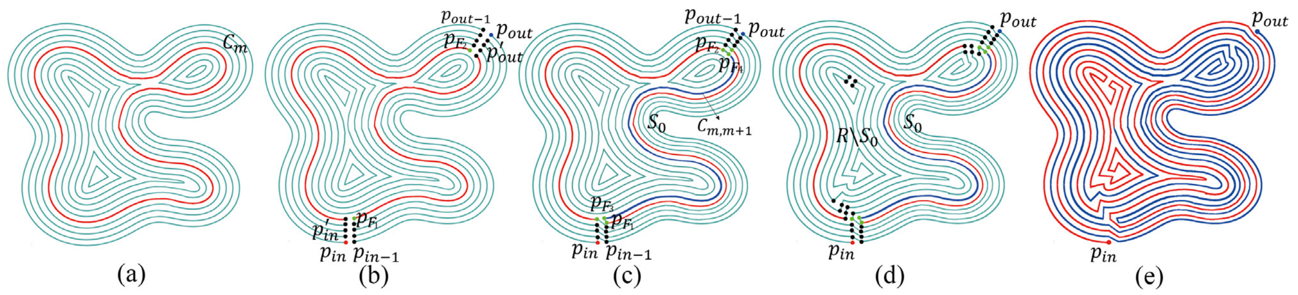

**Fig. 11.** Filling a region with three local minima. (a) Determine the offset curve $C_m$. (b) Determine the connection points in the area $S_0$. (c) Fill the area $S_0$ with a continuous path by adding an extra offset curve segment. (d) Fill the rest part $R\backslash S_0$ with a continuous path. (e) Final result . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

### *4.4.2. Fill a region with multiple local minima*

The method for filling a region with a single local minimum is not suitable for regions with multiple local minima. As illustrated in Fig. 10(e)∼(g), we apply this method to a region with two local minima, where $A$ is the entry point and $D$ is the exit point. Instead of obtaining a continuous path from $A$ to $D$, the path ends at the point $B$ which is not reachable to $D$. In order to get a continuous path from $A$ to $D$, one has to add a curve segment from $B$ to $C$. This idea is feasible for general purpose. In the following, we present an approach for filling a region with multiple local minima and with arbitrary entry point and exit point. The basic approach is to transfer the situation into the case where the region has a single local minimum.

Let $R$ be a region with $K$ ($K \geq 2$) local minima. We want to construct a continuous path in the region with given entry and exit points on the boundary of $R$. The main steps are summarized as follows:

- Compute a series of offset curves $C_1, C_2, \ldots, C_l$ of the boundary curve $\partial R$ (starting from the boundary to the inside of $R$).
- Find the first offset curve $C_m$ where the topology of the offset curve changes (the red curve shown in Fig. 11(a)), and denote by $S_0$ the area bounded by $\partial R$ and $C_m$.
- Fill the area $S_0$ with a continuous path. Firstly we apply the method introduced in the last subsection to construct a continuous path from the entry point $p_{in}$ to some point $p_{F_2}$ on $C_m$, and similarly we construct a continuous path from the exit point $p_{out}$ to some point $p_{F_1}$ on $C_m$ (as shown in Fig. 11(b)). Then we add an extra offset curve segment $C_{m,m+1}$ between $C_m$ and $C_{m+1}$ to connect $p_{F_1}$ and $p_{F_2}$. To make the offset curves more uniform, we assume that the offset curve segment $C_{m,m+1}$ is the center curve of $C_m$ and $C_{m+1}$, and move the corresponding offset curve segments $C_i$ along the normal directions outwards by an amount $\Delta d_i = \frac{i}{m}d$, $i = 1, 2, \ldots, m$. Finally we move $C_{m,m+1}$ by an amount of $d/2$ (as shown in Fig. 11(c)).
- Fill the rest part $R\backslash S_0$ of the region $R$ besides $S_0$ with a continuous path. Here the entry point and exit point are two adjacent points on $C_{m,m+1}$ which can be chosen close to $p_{F_1}$. Since the entry point and the exit point are adjacent, the filling process can be recursively divided into filling areas with a single local minimum (as shown in Fig. 11(d)).
- Connect the filling paths of the two areas $S_0$ and $R\backslash S_0$.

After we obtained a global continuous filling path, we further optimize the smoothness and uniformity of the path by a similar method described in [29].

## 5. Experimental results

In this section, we present a variety of examples to test our path planning algorithm, and comparisons with state-of-the-art methods are also provided. Our experiments were conducted on Ultimate 2+ using FDM (Fused Deposition Modeling) technology. The printing material is PLA with a diameter of 2.85 mm. We use default printer settings, with the nozzle diameter of 0.4 mm, the layer thickness of 0.2 mm, and the maximal nozzle speed of 80 mm per second. The algorithm is implemented in C++ and measured on an Inter(R) Core$^{TM}$ i7-4790 CPU 3.6 GHz with 8 GB RAM.

The most common factors to evaluate the performance of a path generation algorithm are computational time ($ct$), printing time ($pt$), material cost ($m$), number of segments in the path ($seg$), ratio of sharp turns ($st$), ratio of fills (over-fills ($of$) and under-fills ($uf$)) as well as visual effect, where material cost and printing time are the most important factors. We make comparisons on these factors between the four path-generation methods: zigzag method ($Z$), contour-parallel method ($C$), connected Fermat spirals method ($CFS$) [12] and our method ($O$). We use the algorithm in this paper to test some examples (see Fig. 12).

### *5.1. Algorithm performance and printing time*

The main computational cost of our path planning algorithm lies in solving the traveling salesman problem by genetic algorithm. Table 1 summarizes the computational costs (in seconds) for four algorithms—zigzag, contour-parallel, connected Fermat spirals and our method for ten examples. Zigzag algorithm takes the shortest time because of its high computational efficiency, followed by contour-parallel algorithm. Compared with these two classical algorithms, our algorithm and CFS algorithm take more time because more optimization is involved. But our algorithm has a significant improvement over CFS algorithm—only accounts for about 30% of the time of CFS algorithm.

Next we analyze the convergence of solving the TSP using genetic algorithm. Fig. 13 shows the convergence plots for ten examples, where the horizontal axis represents the number of iterations, the vertical axis represents the reciprocal value of fitness functions, and each curve indicates one example. From the picture, we can see that the function values stabilize eventually for all the examples as the number of iterations increases.

Printing time is an important factor in 3D printing since generally it takes dozens of hours to print a model. Columns 6–9 in Table 1 show the actual printing time (in seconds) of one layer for ten examples by the four algorithms. It can be seen that our algorithm takes the least printing time for all the examples, and compared with the other three methods, our method saves about 19%, 15% and 11% printing time on average, respectively.

### *5.2. Material cost*

Porous structures are commonly used in the fields of aerospace and biomedical science in which materials are very expensive.

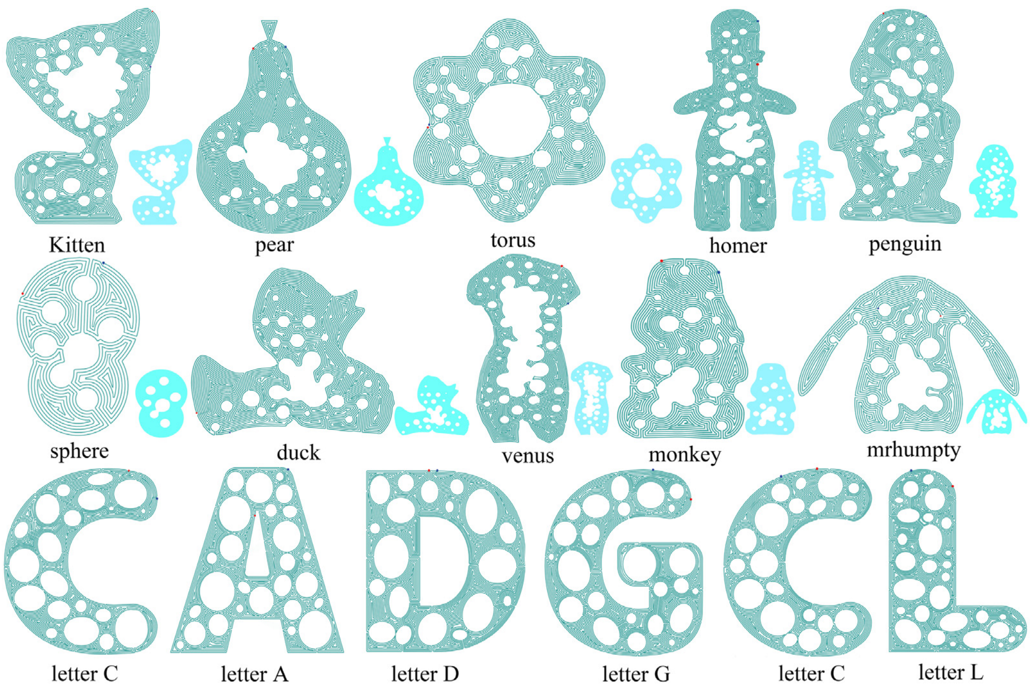

**Fig. 12.** Experimental results with different filling rate and topological complexity that show the feasibility and robustness of our algorithm.

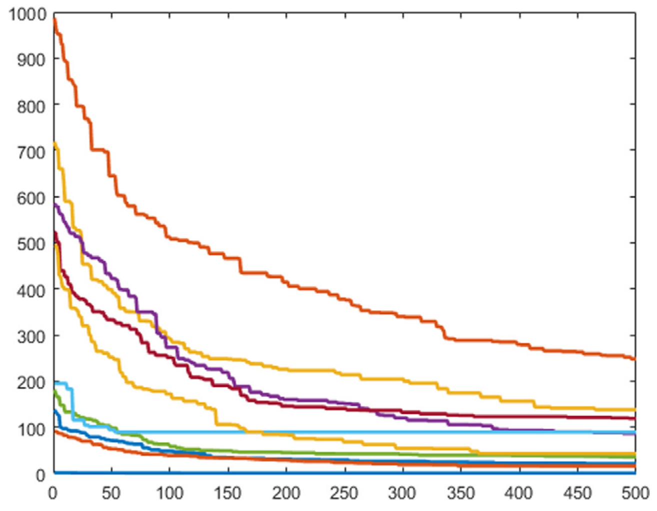

**Fig. 13.** The convergence of genetic algorithm. The horizontal axis represents iteration steps, and the vertical axis represents reciprocal value of fitness function.

**Table 2**
The relative material cost of the four methods for ten examples on average..

| Methods | Zigzag | Contour | CFS | Ours |
|---|---|---|---|---|
| Relative material cost | 2.21 | 1.18 | 1.12 | 1.00 |

In order to meet the requirements of lightweight and minimum cost, it is very important to reduce the material cost. For the four different path filling algorithms mentioned above, we report the average material cost of ten examples for one layer under the same filling rate in columns 10–13 of Table 1. Table 2 shows the relative average material cost of the three methods compared with our method for the ten examples. It can be seen that our algorithm costs least amount of material, and compared with zigzag algorithm, the material cost by our method is less than half of that by the zigzag method. The reason might be that, a large amount of materials are pumped back in the discontinuous sections of zigzag paths and contour-parallel paths. Considering that a model generally consists of hundreds or even thousands of layers, the material saving of printing a model by our method is noticeable.

**Table 1**
Comparison results of four algorithms under algorithm time (at), print time (pt) and length (l)..

| Factors | Computational time (s) | | | | Printing time (s) | | | | Material (mm) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Input* | $Z_{ct}$ | $C_{ct}$ | $CFS_{ct}$ | $O_{ct}$ | $Z_{pt}$ | $C_{pt}$ | $CFS_{pt}$ | $O_{pt}$ | $Z_m$ | $C_m$ | $CFS_m$ | $O_m$ |
| *sphere* | 0.021 | 0.496 | 18.096 | 4.158 | 804 | 783 | 756 | **692** | 2 692 | 1325 | 1243 | **1162** |
| *pear* | 0.020 | 0.565 | 18.769 | 5.115 | 565 | 544 | 524 | **473** | 1 827 | 686 | 646 | **628** |
| *duck* | 0.018 | 0.466 | 15.431 | 3.178 | 683 | 649 | 621 | **558** | 9 390 | 4725 | 4693 | **4492** |
| *kitten* | 0.053 | 0.861 | 20.156 | 5.291 | 587 | 545 | 492 | **412** | 5 537 | 2884 | 2781 | **2473** |
| *torus* | 0.022 | 0.520 | 18.887 | 3.461 | 628 | 606 | 589 | **537** | 2 572 | 1206 | 1193 | **1169** |
| *mrhumpty* | 0.013 | 0.431 | 15.381 | 4.869 | 821 | 796 | 789 | **722** | 12 009 | 5867 | 5893 | **5816** |
| *homer* | 0.041 | 1.056 | 22.469 | 8.305 | 864 | 837 | 796 | **753** | 7 746 | 4297 | 4184 | **4066** |
| *venus* | 0.063 | 1.534 | 64.564 | 26.087 | 562 | 574 | 558 | **542** | 11 231 | 5562 | 5613 | **5196** |
| *penguin* | 0.028 | 0.948 | 19.914 | 8.922 | 786 | 745 | 736 | **619** | 12 693 | 5842 | 5819 | **5623** |
| *monkey* | 0.019 | 0.731 | 15.850 | 4.604 | 829 | 847 | 802 | **725** | 7 592 | 4356 | 4595 | **3913** |

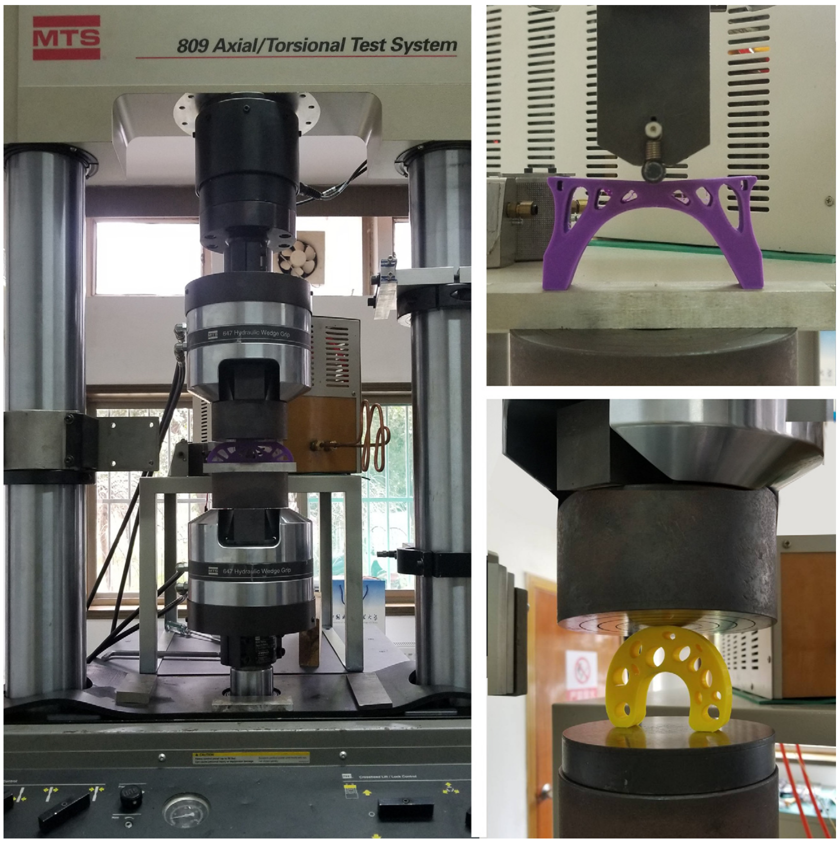

**Fig. 14.** Compression test using MTS809.

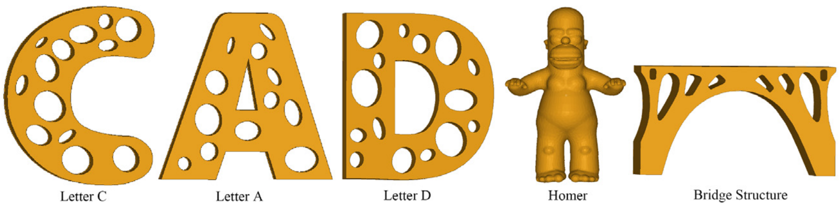

**Fig. 15.** Physical test models.

## 5.3. Structural stability

We also tested the structural stability of five models (as shown in Fig. 15) by the above mentioned four path generation methods, where the bridge model was obtained by the topological optimization technique proposed in [30]. The test was performed on an electromechanical universal testing machine (MTS809) to physically evaluate the strength of the printed model at some point or face as shown in Fig. 14. The crosshead of the machine is controlled to move at a constant speed of 1mm/min to output constant compression force. We recorded the maximum affordable stress for each model. Table 3 shows the weight, the maximum stress and the ratio of the maximum stress over the weight for each model. In each box of the Z's columns, the two numbers correspond to the maximum stresses in two special printing directions by the zigzag method—the maximum number is the stress of the direction that is parallel to the machine movement direction, and the minimum number is the stress of the direction that is perpendicular to the machine movement direction.

From Table 3, we can see that the maximum stress for zigzag algorithm varies greatly according to the printing direction, or the direction how the force is applied, and the stress can be as low as only 50%–70% of the stress by our method in the worst direction. On the other hand, our method produces the largest stress among the other three methods. Most noticeably, the load capacities of per unit weight by our method are much larger than those by the other methods.

## 5.4. Path continuity

Global continuity of the filling path reduces printing time and material cost. CFS and our algorithms are globally continuous while the other two algorithms produce filling paths consisting of large number of segments as shown in columns 2–3 of Table 4.

## 5.5. Sharp turns

Near a sharp turn the print head will slow down at the apex and then accelerate away from the apex. Uneven printing velocity can cause time wasting and affect printing quality. Therefore, it is necessary to reduce the ratio of sharp turns to ensure printing accuracy and efficiency.

Let $p_1, p_2, \ldots, p_n$ be uniform sampling points on a filling path. A sharp turn is the point $p_i$ on the filling path where the *turning angle* between $p_{i-1}p_i$ and $p_ip_{i+1}$ is less than some threshold $\delta$. To

**Table 3**
Stress-to-weight ratios of five models by the four path generation methods.

| Models | Letter C | | | | Letter A | | | | Letter D | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Method* | *Z* | *C* | *CFS* | *O* | *Z* | *C* | *CFS* | *O* | *Z* | *C* | *CFS* | *O* |
| Weight (g) | 17.35 19.23 | 15.81 | 14.32 | 13.18 | 17.16 15.28 | 14.85 | 14.21 | 13.12 | 14.54 13.17 | 12.76 | 12.31 | 11.68 |
| Stress (N) | 1433 1108 | 1307 | 1294 | 1338 | 1749 1003 | 1291 | 1235 | 1390 | 1327 847 | 912 | 931 | 1124 |
| Ratio | 82.59 57.62 | 82.67 | 90.36 | **101.52** | 101.92 65.64 | 86.94 | 86.91 | **105.95** | 91.27 64.31 | 71.47 | 75.63 | **96.23** |
| Models | Homer | | | | Bridge structure | | | | | | | |
| *Method* | *Z* | *C* | *CFS* | *O* | *Z* | *C* | *CFS* | *O* | | | | |
| Weight (g) | 39.34 36.23 | 32.58 | 30.41 | 25.97 | 18.56 15.94 | 17.58 | 16.94 | 14.87 | | | | |
| Stress (N) | 1840 1209 | 1631 | 1693 | 1704 | 278 194 | 254 | 242 | 268 | | | | |
| Ratio | 46.77 33.37 | 50.06 | 55.67 | **65.61** | 14.98 12.17 | 14.45 | 14.28 | **18.02** | | | | |

**Table 4**
Number of path segments (seg) and permillage of sharp turn (st)..

| *Input* | $Z_{seg}$ | $C_{seg}$ | $Z_{st}$ | $C_{st}$ | $CFS_{st}$ | $O_{st}$ |
|---|---|---|---|---|---|---|
| *sphere* | 84 | 113 | 9.32% | 4.51% | 4.31% | **4.08%** |
| *pear* | 53 | 107 | 8.24% | 5.82% | 5.54% | **5.17%** |
| *duck* | 86 | 112 | 7.65% | 4.19% | 4.12% | **3.75%** |
| *kitten* | 175 | 114 | 8.81% | 5.89% | 5.62% | **5.03%** |
| *torus* | 158 | 86 | 7.29% | 4.58% | 4.13% | **3.64%** |
| *mrhumpty* | 92 | 70 | 9.13% | 5.27% | 5.04% | **4.87%** |
| *homer* | 126 | 98 | 9.82% | 4.83% | 4.97% | **3.78%** |
| *venus* | 112 | 142 | 9.61% | 4.15% | 4.25% | **4.09%** |
| *penguin* | 178 | 148 | 8.07% | 4.62% | **4.39%** | 4.46% |
| *monkey* | 76 | 63 | 7.96% | 3.73% | 3.83% | **3.12%** |

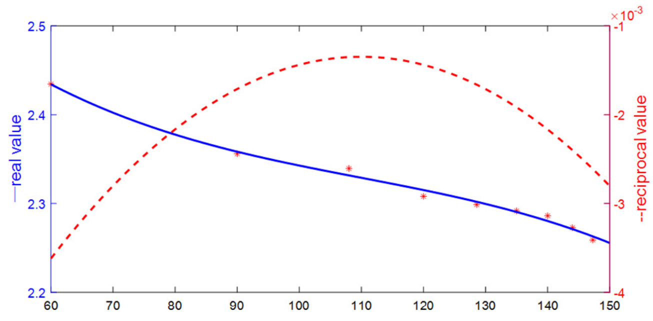

**Fig. 16.** Printing time vs. turning angles . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

determine the appropriate threshold value $\delta$, we count the actual printing time to print regular polygons with different sides. The result is plotted in Fig. 16, where the horizontal axis represents the inner angles of the regular polygons, and the vertical axis represents the printing time. As we can see that the printing time decreases as the angle increases. We fit the data with a cubic polynomial $f(\theta)$ (blue curve) and set $f'(\theta) = 0$ to obtain the threshold value $\delta \approx 110°$.

For all the points on the filling path, we calculate the turning angles and check if they are less than the threshold $\delta = 110°$. If the angle is less than $\delta$, then the point is regarded as a sharp turn. Columns 4–7 in Table 4 list the sharp turn ratios for different methods. Again our algorithm produces least number of sharp turns except for one example.

## 5.6. Over-fills and under-fills

Over-fills and under-fills influence printing quality and increase material wasting. In order to quantitatively calculate the ratios of over-fills and under-fills, we think of a printing path as a curve with a preset width (nozzle diameter). The path will intersect at the over-fills areas and generate gaps at the under-fills areas. Calculate the proportion of the intersection area and gaps to obtain the ratios of over-fills and under-fills. The histograms in Fig. 17(a) and (b) show the ratios of under-fills and over-fills for the four algorithms on ten examples. It can be seen that the zigzag fill pattern has the least ratio of under-fills, but its over-fills ratio is the highest. Contour-parallel algorithm causes filling unevenly due to small pieces produced in the offset computing. CFS fill patterns tend to increase more gaps and under-fills. Our filling algorithm produces the least over-fills among the four methods and less under-fills than the CFS method and contour-parallel method on average.

## 5.7. Visual effect

The visual effect is a comprehensive embodiment of many factors. To evaluate the visual effect of different methods, we made a user-study. We invited 20 users to make scores about the visual effect for ten examples. 1 stands for the poorest visual effect, and 10 represents the best visual effect. The average rating results are shown in Fig. 18. From the histogram, it can be concluded that our method produces the best visual effect for all the examples.

Fig. 19 illustrates the generating paths of two models by the four methods, while Fig. 20 shows the actual printing results of two models. As we can see from the results, zigzag paths are uniform, but have serrated parts at the boundary due to staircase effect. As for the contour-parallel algorithm, the staircase effect is improved. However, printing material may be drawn out of the paths due to the discontinuity of the paths for these two methods. CFS method produces more under-fills or gaps. Comparatively our method generates more uniform and smooth paths and can effectively reduce gaps and visible artifacts.

## 5.8. Limitations

Our algorithm suffers from some deficiencies since it preserves the weakness of contour-parallel path filling methods inherently, that is, there are still some over-fills and under-fills in the final printing results although our method reduces the proportion

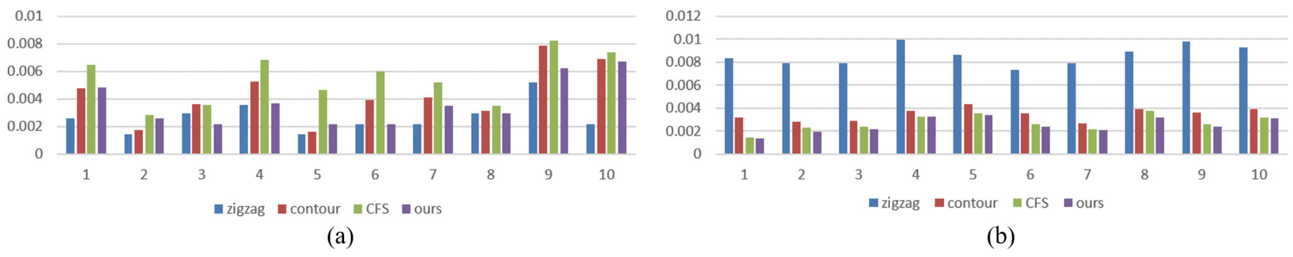

**Fig. 17.** Ratios of under-fills (a) and over-fills (b) on ten examples.

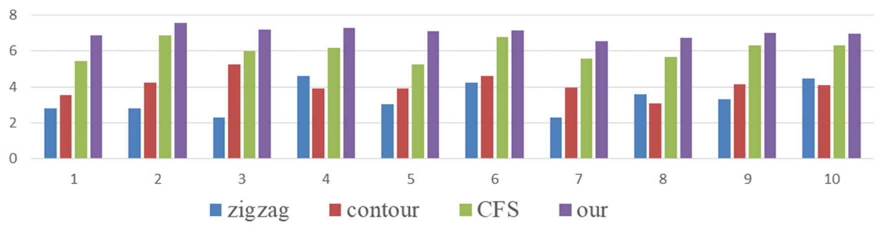

**Fig. 18.** The scores of visual effect for ten examples.

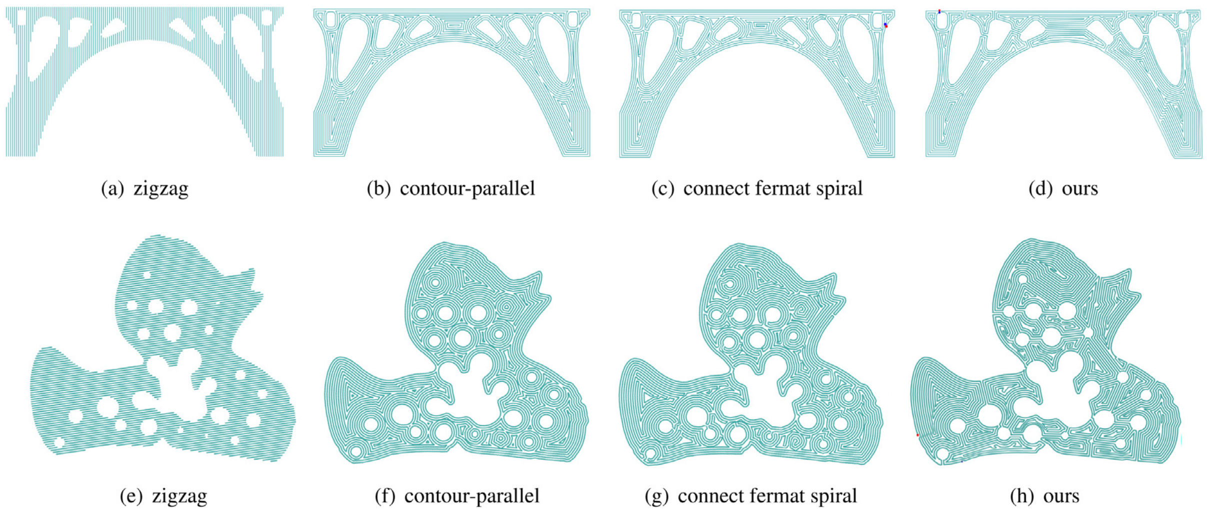

**Fig. 19.** Paths of two models generated by four path-planning methods.

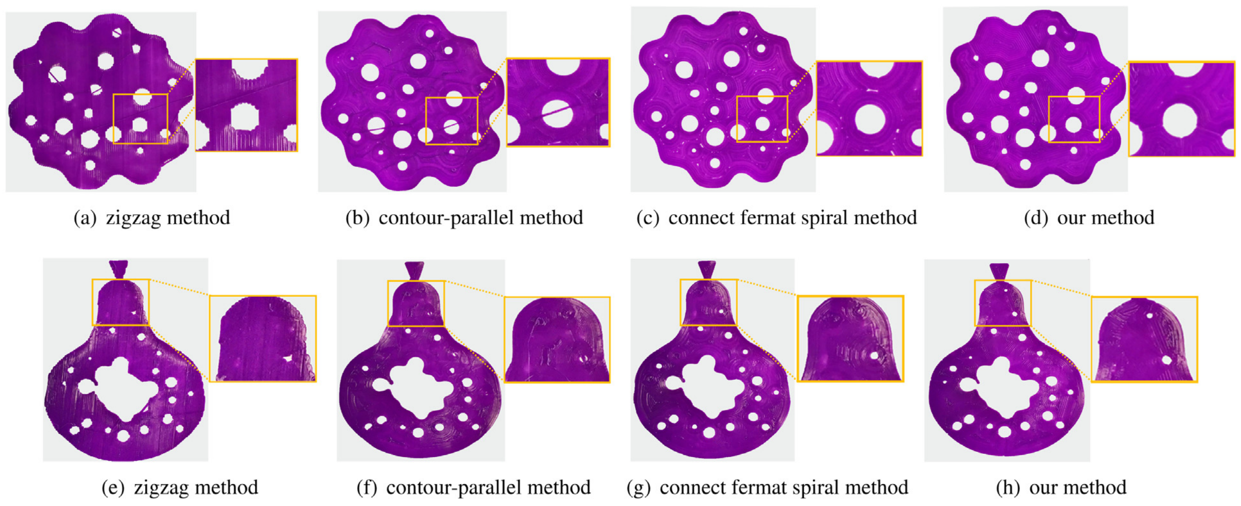
(a) zigzag method
(b) contour-parallel method
(c) connect fermat spiral method
(d) our method
(e) zigzag method
(f) contour-parallel method
(g) connect fermat spiral method
(h) our method

**Fig. 20.** Printing results of two models.

of over-fills and under-fills compared with other methods. In addition, when solving the TSP problem using genetic algorithm, the algorithm efficiency decreases as the number of holes in the porous structure increases. Fortunately, we can divide each slice into subregions for parallel computation to improve the efficiency of the algorithm.

## 6. Conclusion

In this paper, we present an approach to generate printing paths for a type of porous structures which contain lots of small holes. Our work adopts the divide-and-conquer strategy. Each slice of a porous structure is divided into subregions by solving a salesman traveling problem using genetic algorithm, and then each subregion is filled with a Fermat spiral curve. We provided many examples to demonstrate the effectiveness and superiority of our method and comparisons with other state-of-the-art methods are also provided. Experimental results show that our algorithm outperforms others methods in terms of material cost, printing time and structural stability.

In the future work, we will investigate topology optimization of porous structures which is a very important research topic. Constructing more uniform printing paths using more efficient algorithm is worthy of further study. Generalizing the work to fill printing paths on surfaces is also very interesting.

## References

[1] Park S, Choi B. Tool-path planning for direction-parallel area milling. Comput Aided Des 2000;32(1):17–25. http://dx.doi.org/10.1016/S0010-4485(99)00080-9, URL http://www.sciencedirect.com/science/article/pii/S0010448599000809.
[2] Rajan V, Srinivasan V, Tarabanis KA. The optimal zigzag direction for filling a two dimensional region. Rapid Prototyping J 2001;7(5):231–41. http://dx.doi.org/10.1108/13552540110410431, URL https://doi.org/10.1108/13552540110410431.
[3] Kim H-C. Tool path generation and modification for constant cutting forces in direction parallel milling. Int J Adv Manuf Technol 2011;52(9):937–47. http://dx.doi.org/10.1007/s00170-010-2790-4.
[4] Jin Y-a, He Y, Xue G-h, Fu J-z. A parallel-based path generation method for fused deposition modeling. Int J Adv Manuf Technol 2015;77(5–8):927–37.
[5] Farouki R, Koenig T, Tarabanis K, Korein J, Batchelder J. Path planning with offset curves for layered fabrication processes. J Manuf Syst 1995;14(5):355–68. http://dx.doi.org/10.1016/0278-6125(95)98872-4, URL http://www.sciencedirect.com/science/article/pii/0278612595988724.
[6] Yang Y, Loh H, Fuh J, Wang Y. Equidistant path generation for improving scanning efficiency in layered manufacturing. Rapid Prototyping J 2002;8(1):30–7. http://dx.doi.org/10.1108/13552540210413284, URL https://doi.org/10.1108/13552540210413284.
[7] Abdullah H, Ramli R, Wahab DA. Tool path length optimization of contour parallel milling based on modified ant colony optimization. Int J Adv Manuf Technol 2017;92(1):1263–76. http://dx.doi.org/10.1007/s00170-017-0193-5.
[8] Nair SH, Sinha A, Vachhani L. Hilbert's Space-filling Curve for Regions with Holes, CoRR abs/1709.02938, URL arXiv:1709.02938, 2017.
[9] Fleming C, Walker S, Branyan C, Nicolai A, Hollinger G, Mengüç Y. Toolpath Planning for Continuous Extrusion Additive Manufacturing.
[10] Held M, Spielberger C. A smooth spiral tool path for high speed machining of 2d pockets. Comput Aided Des 2009;41(7):539–50. http://dx.doi.org/10.1016/j.cad.2009.04.002, URL http://www.sciencedirect.com/science/article/pii/S0010448509001031.
[11] Held M, Spielberger C. Improved spiral high-speed machining of multiply-connected pockets. Comput-Aided Des Appl 2014;11(3):346–57.
[12] Zhao H, Gu F, Huang Q-X, Garcia J, Chen Y, Tu C, Benes B, Zhang H, Cohen-Or D, Chen B. Connected fermat spirals for layered fabrication. ACM Trans Graph 2016;35(4):100:1–100:10. http://dx.doi.org/10.1145/2897824.2925958, URL http://doi.acm.org/10.1145/2897824.2925958.
[13] Jin G, Li W, Gao L. An adaptive process planning approach of rapid prototyping and manufacturing. Robot Comput-Integr Manuf 2013;29(1):23–38.
[14] Ding D, Pan ZS, Cuiuri D, Li H. A tool-path generation strategy for wire and arc additive manufacturing. Int J Adv Manuf Technol 2014;73(1):173–83. http://dx.doi.org/10.1007/s00170-014-5808-5.
[15] Ozbolat IT, Khoda A. Design of a new parametric path plan for additive manufacturing of hollow porous structures with functionally graded materials. J Comput Inf Sci Eng 2014;14(4):041005.
[16] Lu L, Sharf A, Zhao H, Wei Y, Fan Q, Chen X, Savoye Y, Tu C, Cohen-Or D, Chen B. Build-to-last: strength to weight 3d printed objects. ACM Trans Graph 2014;33(4):97:1–97:10. http://dx.doi.org/10.1145/2601097.2601168, URL http://doi.acm.org/10.1145/2601097.2601168.
[17] Wu J, Aage N, Westermann R, Sigmund O. Infill Optimization for Additive Manufacturing - Approaching Bone-like Porous Structures, CoRR abs/1608.04366, arXiv:1608.04366, 2016.
[18] Ying J, Lu L, Tian L, Yan X, Chen B. Anisotropic porous structure modelling for 3d printed objects. Comput Graph 2018;70:157–64. http://dx.doi.org/10.1016/j.cag.2017.07.008, URL http://www.sciencedirect.com/science/article/pii/S0097849317301012, CAD/Graphics 2017.
[19] Bloomenthal J, Wyvill B, editors. Introduction to Implicit Surfaces. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 1997.
[20] Song Y, Yang Z, Liu Y, Deng J. Function representation based slicer for 3d printing. Comput Aided Geom Design 2018;62:276–93.
[21] Muraki S. Volumetric shape description of range data using blobby model. SIGGRAPH Comput Graph 1991;25(4):227–35. http://dx.doi.org/10.1145/127719.122743, URL http://doi.acm.org/10.1145/127719.122743.
[22] Song X, Chen F. Adaptive surface reconstruction based on tensor product algebraic splines. Numer Math Theory Methods Appl 2009;2(1):90–9.
[23] Lloyd S. Least squares quantization in PCM. IEEE Trans Inform Theory 1982;28(2):129–37. http://dx.doi.org/10.1109/TIT.1982.1056489.
[24] Bendsøe MP. Topology optimization. Springer; 2009.
[25] Adams D, Turner C. An implicit slicing method for additive manufacturing processes. Virtual Phys Prototyping 2018;13(1):2–7.
[26] Hoff III KE, Keyser J, Lin M, Manocha D, Culver T. Fast computation of generalized voronoi diagrams using graphics hardware. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '99, New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.; 1999, p. 277–86. http://dx.doi.org/10.1145/311535.311567.
[27] Rudolph G. Convergence analysis of canonical genetic algorithms. IEEE Trans Neural Netw 1994;5(1):96–101.
[28] Johnson A. Clipper - an open source freeware library for clipping and offsetting lines and polygons. 2010-2014.
[29] Chen Z, Shen Z, Guo J, Cao J, Zeng X. Line drawing for 3d printing. Comput Graph 2017;66:85–92. http://dx.doi.org/10.1016/j.cag.2017.05.019, URL http://www.sciencedirect.com/science/article/pii/S0097849317300687, Shape Modeling International 2017.
[30] Sigmund O. A 99 line topology optimization code written in matlab. Struct Multidiscip Optim 2001;21(2):120–7.