3D Printing Path Optimization Strategy Based on Dijkstra's Algorithm
—The Contour Slicing Method with Eulerian Loop Optimization as an Example

Zhe Lai¹, Yalong Mao², Lukui Huang³, Hao Zheng⁴
¹School of Architecture, South China University of Technology, Guangzhou, China
²Architectural Design & Research Institute, State Key Laboratory of Subtropical Building Science, School of Architecture, South China University of Technology, Guangzhou, China
³Business School, Thammasat University, Bangkok, Thailand
⁴Department of Architecture and Civil Engineering, City University of Hong Kong, HKSAR, China

Abstract
With the expeditious evolution of 3D printing technology, there is a growing need to improve its efficiency and quality. In the context of 3D printing of concrete in architectural applications, conventional slicing software, originally devised for PLA materials, has proved inadequate in tackling issues such as accumulation, gaps, and backfilling. This paper presents a 3D printing path optimization strategy that utilizes the Dijkstra algorithm for architects in Rhino and Grasshopper environments. This approach provides the flexibility to adjust path direction and layers freely, and reduces sedimentation problems without the need for support materials. To validate the effectiveness of the system, this paper conducted tests on four typical 3D printing tasks. The results show that compared to the Cura system, the optimisation rates of the system for the four task types were 23.21%, 57.14%, 50% and 40% respectively. The average response time for the four types of 3D printing tasks was approximately 0.1211 seconds. Additionally, it enhances the interaction between architects and 3D printing devices, which is currently lacking in the field. The proposed method offers greater possibilities and customization for the application of 3D printing in construction.

Keywords: Dijkstra's algorithm, 3D printing, contour slicing method, path optimization, Euler circuit.

Highlights
- This article proposes an automated method for optimizing the 3D printing project path, which effectively reduces issues such as stacking, gaps, and refilling during printing.
- A user-friendly interface system is provided for interaction between architects and 3D printing equipment.
- The results show that using Dijkstra's algorithm for path optimization reduces the consumption of printing materials.
- The system response time and speed were improved for various tasks.

Introduction
The rapid development of 3D printing technology has led to a pressing need to enhance the efficiency and quality of 3D printing, with path optimization being a crucial means of achieving this objective. In this paper, using the contour slicing method optimization strategy as an example, we propose an adaptive path optimization strategy based on a genetic algorithm-Dijkstra algorithm for 3D printing. Previous research has explored to deploy machine learning and genetic algorithms to improve the efficiency and quality in the digital assembly of 3D printing. Ahmad Bilal, for instance, employed genetic algorithms in tandem with machine learning technology to develop digital 3D printing in support of assembly operations. This technology facilitates the design, simulation, and optimization of the intricacies of 3D printing [1]. Zangaro Francesco utilized machine learning techniques to optimize the paths on the same plane for 3D printing according to the selected properties of the components and the intelligent construction environment [2]. Research on layer efficiency mainly aims to improve time efficiency [3], such as real-time adaptive slicing for FDM technology [4]. Singhal et al. proposed the "adaptive slicing based on CAD models" method to facilitate the coordination between manufacturing time and surface quality [5]. Research on path planning in 3D printing primarily focuses on path generation and optimization methods such as scan filling [6-9]. Rajan and Misra proposed a bidirectional parallel scan filling algorithm-Zig Zag- which improves unidirectional scanning and is currently the most basic filling method in 3D printing [10-12]. During path planning, if there are too many empty travel moves in the scan path , it can lead to repeated start-stop of the printer nozzle, frequent filament retractions, and affect the quality of the printed object. To address this issue, Yang and Huang Xuemei proposed a partitioned scan filling algorithm based on bidirectional scanning [13]. To reduce the impact of the "staircase effect" on dimensional accuracy, Wurikaix et al. proposed an adaptive spacing algorithm based on the relationship between processing scan spacing and part dimensional accuracy [14]. Zhao et al. [15] studied the scanning of complex sectional contours and sub-regions, optimizing the scan paths between regions and extreme positions.
within sub-regions, thereby improving the manufacturing efficiency.

Building on these prior efforts, this paper employs the latest genetic algorithm to further optimize the path planning for 3D printing, and shows great improvement in efficiency and quality of 3D printing.

**Isoline Slicing Method Optimized with Eulerian Circuit**

The isoline slicing method is a common 3D printing path planning algorithm that decomposes a model into multiple horizontal planes and generates isolines on each plane. These isolines are then connected in a certain order to form a printing path. However, imperfect path planning in isoline slicing can lead to issues such as path overlap, gaps, and excessive filling between layers. In optimizing the isoline slicing method, it can be applied to the optimization of filling paths, reducing unnecessary stacking between layers. Therefore, the optimization of isoline slicing path is crucial in improving the efficiency and quality of 3D printing.

An Eulerian circuit denotes a type of circuit that traverses every edge of a given graph exactly once, with the starting and ending points being identical. Within the context of isoline slicing, optimization of the filling path can be attained through the utilization of Dijkstra's algorithm. The resultant outcome constitutes a shortest path, whereby the starting and ending points are identical, and every edge is traversed exactly once. This implies that the path can be deemed as an Eulerian circuit.

In the practice of 3D printing, implementation of Eulerian circuit optimization onto the isoline slicing technique presents the potential to effectively diminish the expenses of both time and materials. Conventional isoline slicing procedures generate a substantial amount of layers, each equipped with a distinct filling path. By integrating Eulerian circuit optimization, these filling paths can be efficiently interconnected, thereby decreasing the pause interval between successive horizontal layers and minimizing the displacement time of the printing head. Such a process effectively mitigates the wastage between filling paths, culminating in heightened efficiency and enhanced printing outcomes.

**3D Printing Based on Grasshopper**

Grasshopper is a visual programming tool for designers and architects that can be used in conjunction with CAD software such as Rhino. Its flexibility and extensibility allow users to design more complex structures in the field of 3D printing.

In this implementation, the programming was done using C# language and visual programming was achieved through Grasshopper. After algorithm optimization and comparison with the printing path of the traditional printing system Cura, it was found that the path optimization strategy based on Dijkstra's algorithm can effectively avoid problems such as path overlapping, gaps, and filling, while improving printing efficiency while ensuring quality.

This paper also presents how to use Grasshopper to generate the optimal printing path, thereby minimizing printing time and material waste while ensuring printing quality.

**Dijkstra's Shortest Path Algorithm**

In 3D printing, commonly used algorithms for optimizing the optimal path include Genetic Algorithm (GA), Traveling Salesman Problem (TSP) algorithm, Simulated Annealing (SA) algorithm, A* (A star) algorithm, and Dijkstra's algorithm.

Dijkstra's algorithm is a classic single-source shortest path algorithm used to calculate the shortest path from the starting point to all other nodes. It is often used for path planning in graphics. Dijkstra's algorithm considers each node as a vertex in the graph and each edge as a path between vertices, and then calculates the shortest path for path planning.

The following is the general expression of Dijkstra's algorithm:

Let $G = (V, E)$ be a weighted directed graph, where $V$ is the set of nodes and $E$ is the set of edges.

Define the starting point $s$, the end point $t$, the edge weight $w$, and the shortest path length $d(v)$.

$s$ is the starting point, and $t$ is the end point.

$w(e)$ represents the weight of edge $e$.

$d(v)$ represents the shortest path length from the starting point $s$ to node $v$.

Use the set $S$ to represent the shortest path from the starting point $s$ to the nodes in set $S$.

**Algorithm flow:**

*Initialization*: Initialize the distance from the starting point $s$ to all nodes to infinity and the distance from the starting point to itself to 0.

*Find the node with the shortest distance*: In set $V - S$, select the node $u$ closest to the starting point $s$ and add it to set $S$.

*Update the distance value*: For all nodes $v$ in set $V - S$, if there is an edge $e = (u, v)$ such that $d(u) + w(e) < d(v)$, then update $d(v) = d(u) + w(e)$.

Repeat steps 2 and 3 until all nodes are added to set $S$. Determine the shortest path from the starting point $s$ to the end point $t$ based on $d(v)$.

**Project Objective**

This project aims to use the Dijkstra algorithm as a research method to optimize the 3D printing of Euler circuits in contour slicing as the main research problem, and to propose a practical new system based on this. Finally, the effectiveness of the system is verified by comparing the differences in optimization rate, response time, and simulation effect between the traditional 3D printing system Cura and this system.
Methods

Workflow

The essence of concrete 3D printing is layer-by-layer manufacturing, and its key technology is the processing of the three-dimensional model of the concrete component. Path optimization directly affects the forming quality and efficiency of the final component. Due to its characteristic of stacked printing, the forming quality and printing time of each layer have a huge impact on the overall forming quality and printing efficiency of the final component. The overall process of concrete 3D printing can be divided into five parts:

(1) Three-dimensional modeling of concrete components;
(2) Slicing the 3D model using Grasshopper;
(3) Optimizing the printing path for the sliced model;
(4) Writing the corresponding G-code;
(5) Actual printing. Figure 2 shows the process flowchart for concrete 3D printing.

Path Optimization Algorithm

Reducing the number of nozzle lifts and printing travel in 3D printing of concrete is the two main issues in path optimization. Reducing the number of nozzle lifts can be approximately divided into the "one-stroke problem", that is, whether a shape on a plane composed of line segments can be drawn in one stroke, such that a shape is not drawn repeatedly on each line segment. If the shape is an Eulerian graph, then it can be drawn in one stroke [3]. However, not every geometric shape of a concrete component is an Eulerian graph. To determine which Eulerian graph the printing nozzle should traverse first after finding the Eulerian graph, this paper introduces the Dijkstra algorithm to optimize the lifting order of the printing nozzle and reduce the printing travel. Figure 2 shows the schematic diagram of the Eulerian circuit in this paper.

Implementation Process and Code

To begin with, we need to build a model in Grasshopper. Suppose we want to print a simple object like the one shown in the figure below, which is composed of a group of adjacent triangles. Next, we need to convert the model into a mesh and use the "Mesh Edges" component to convert the mesh into a connection relationship between nodes.

In the above definition, we first convert the model into a mesh, and then use the "Mesh Edges" component to convert the mesh into a connection relationship between nodes. In the above definition, we first convert the model into a mesh, and then use the "Mesh Edges" component to convert the mesh into a connection relationship between nodes. Finally, we input the distance matrix into the "Dijkstra" component to find the shortest path from the starting point to the ending point.

The following is the definition of the "Dijkstra" path planning component:

(1) Customizing a path planning component named "Dijkstra" in Grasshopper, which can be used to execute the Dijkstra algorithm to find the shortest path.

Path Overall Optimization

The basic idea of generating optimized paths using the Dijkstra algorithm can be summarized as following: Firstly, in Grasshopper, the 3D model is sliced to generate contours. Then, using an optimized contour slicing method, each layer is treated as a node, and the distance between nodes can be represented as the length of the backfill path. The backfill path between adjacent layers can be treated as an edge, and the weight of the edge can be represented as the length of the backfill path. The first layer is defined as the starting point, and the last layer is defined as the endpoint. The details are presents as below:

(1) Define the starting point and the end point.
(2) Create a graph, with the grid as nodes and the connections between nodes as edges. Set all edge weights to 1 or the distance between nodes.
(3) Create a priority queue to store unexplored nodes.
(4) Initialize the distances of all nodes to infinity, set the distance of the starting node to 0, and add the starting node to the priority queue.
(5) At each iteration, remove the node with the shortest distance from the priority queue, mark it as explored, and add its neighboring nodes to the priority queue. Update the distance values of the neighboring nodes.
(6) When the end point is removed from the priority queue, the algorithm ends. Output the shortest path from the starting point to the end point.
(2) This component requires input of the connection relationship between nodes, which can be generated using the "Mesh Edges" component in Grasshopper. The connection relationship can be input by defining a two-dimensional array, where each element represents the distance between nodes. After inputting, the component will output the shortest path from the starting point to the ending point.

(3) The specific process and code for optimizing the path with Euler loop are shown in Figure 3.

### Table 1 The assembly information

<table>
<thead>
<tr>
<th>No.</th>
<th>The proposed strategy distance/mm</th>
<th>Cura (Software) distance/mm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>66186.61</td>
<td>861961.65</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>105</td>
</tr>
<tr>
<td>3</td>
<td>20.76</td>
<td>41.23</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>10</td>
</tr>
</tbody>
</table>

(4) Determine the optimal backfill path based on the shortest path from the starting point to each node, and connect them in the order of the paths. In contour slicing, the shortest path can be represented as a series of adjacent nodes, and the layers represented by these nodes form an Euler circuit. The Euler circuit can be optimized to minimize the number of stacking layers.

(5) Finally, an optimized printing path is formed, thereby improving the efficiency and quality of 3D printing.

### Print Experiments and Results

To verify the practicality of this method in construction projects, this paper conducted performance tests by outputting optimized 3D printing Gcode. To visually reflect the performance of the algorithm, this paper compared it with the commonly used traditional 3D printing slicing software, Cura. The testing process mainly includes implementing automatic planning function, data information management function, and task response time.

#### Comparison of path optimization

On the platform, this paper tested the feasibility of two systems with different slicing path planning based on the actual situation of four sets of data in the following table. The assembly information is shown in Table 1, and the feasibility results of the system planning are shown in Figure 4.

![Figure 3: Flowchart and Code for Euler Circuit Optimization Path](image)

![Figure 4: Test results of planning function](image)

From the overall test results in Figure 4, there is a clear difference between the two systems in terms of automatic planning effectiveness. In Figure 5, the optimization ratios of the digital 3D printing path planning in this paper are all ideal. Without changing the nozzle running speed, compared with the Cura system, the software path optimization in this paper achieves a 23.21% improvement in total distance, a 57.14% improvement in nozzle lifting times, a 50% improvement in nozzle backfilling, and a 40% improvement in printing overlap optimization.

The result shows that the 3D printing path optimization based on the Dijkstra algorithm can effectively improve the efficiency of automatic planning in 3D printing. Compared with the Cura 3D printing system Cura, the latter is more affected by the original path. When the printing distance increases, the proposed system is...
superior to the traditional system in terms of printing material deposition and backfilling effect.

**Comparison of Task Response Time**

Another performance metric used to evaluate the effectiveness of a 3D printing system in this paper is task response time. This study compared the response time of the two systems in handling different printing tasks. The results are shown in Figure 4.

Moreover, the intersections between layers in the Z direction are presented as oblique angles.

After verification, the optimized path did indeed improve issues such as path overlap, gaps, and backfilling.

**Conclusion**

(1) Compared with the Cura system, the software path optimization of the proposed system in terms of total travel distance was improved by 23.21%, nozzle lifting optimization by 57.14%, nozzle retraction optimization by 50%, and printing overlap optimization by 40%.

(2) The response times for the four types of 3D printing tasks were 0.0638s, 0.0976s, 0.1327s, and 0.1901s, with an average response time of approximately 0.1211s. The experimental results show that, without changing the nozzle travel speed, the proposed algorithm effectively reduces the problems of stacking, gaps, and retraction that may occur during 3D printing operations, improves the system's response speed, and enhances the weak interaction between architects and 3D printing equipment through a user-friendly interactive operating system.

However, it should be noted that the Dijkstra algorithm for path optimization is not the best method available. In future research, a comparative study of the A* algorithm and the SA algorithm will be conducted, and further exploration and improvement will be made to improve the efficiency of 3D printing in architecture.

**Acknowledgement**

This work was supported by National Social Science Foundation, "Research on Protection and Renewal Mechanism, Methods and Improvement Countermeasures of Traditional Villages Based on Fractal Thinking"(21BGL258); 2022 Independent Research Project of the State Key Laboratory of Subtropical Architecture Science, "Research on Renewal
Mechanism and Fractal Iteration Method of Existing Industrial Buildings in Subtropical Region Based on Urban Double Renovation Concept.

References


