As-Continuous-As-Possible Extrusion Fabrication of Surface Models

FANCHAO ZHONG, Shandong University
YONGLAI XU, Shandong University
HAISEN ZHAO, IST Austria, Shandong University and University of Washington
LIN LU, Shandong University

We propose a novel computational framework for optimizing the toolpath continuity in fabricating surface models on an extrusion-based 3D printer. Toolpath continuity has been a critical issue for extrusion-based fabrications that affects both quality and efficiency. Transfer moves cause non-smooth or bumpy surfaces and get worse for materials with large inertia like clay. For surface models, the effects of continuity are even more severe, in terms of surface quality and model stability. In this paper, we introduce an original criterion "one-path-patch" (OPP), for representing a shell surface patch that can be traversed in one path considering fabrication constraints. We study the properties of an OPP and the merging operations for OPPs, and propose a bottom-up OPP merging procedure for decomposing the given shell surface into a minimal number of OPPs and generating the "as-continuous-as-possible" (ACAP) toolpath. Furthermore, we customize the path planning algorithm with a curved layer printing scheme, which reduces the staircase defect and improves the toolpath continuity via possibly connecting multiple segments. We evaluate the ACAP algorithm for both ceramic and thermoplastic materials, and results demonstrate that it improves the fabrication of surface models in both surface quality and efficiency.

1 INTRODUCTION

Surface or shell models are widely used in structural design, being efficient in shape presentation and possessing featured functionalities like lightweight and effective thermal conductivity. Potteries in the shell form have been developed since the Stone Age. In the context of additive manufacturing (AM), shells are cost-effective for both materials and fabrication time compared to solid ones. In this paper, we focus on the surface model, particularly a thin shell with the thickness of a single path. It can be either open or closed, and therefore segments exist in layers for open surface models.

Continuity of the toolpath is one of the most fundamental problems in material extrusion-based AM. Continuity of the nozzle’s movement and material extrusion directly affects the surface quality, model stability, and fabrication efficiency. For extrusion fabrication of surface models, the toolpath continuity plays a more critical role than solid models. Besides the surface quality and printing efficiency, transfer moves induce extra forces to the printed shell surface, weakening the stability, and the model may sag or collapse along with the accumulation of the forces.

Especially, fabricating surface models with clay is getting popular thanks to the rapid progress on the ceramics printing techniques. The most feasible and cost-effective technique for 3D printing clay is direct ink writing (DIW), which shares the same architecture.
Curved layers manufacturing is regarded as an effective method for removing the staircase defects [Etienne et al. 2019] and improving the strength through aligning filaments along the directions with large stresses [Fang et al. 2020]. For surface models, curved layers have unique advantages that multiple components may be printed in a connected toolpath.

Adopting the curved layers scheme for Cartesian 3D printing faces two more constraints from fabrication besides continuity. First, the layer thickness is adjustable but bounded by the extrusion amount. Second, the slope of the curved toolpath cannot be too steep, as the collision between the nozzle and printed model must be considered. Considering the fabrication constraints, the problem can be regarded as a Precedence Constrained Minimum Path Cover Problem (PC-MPC), which is equivalent to a classical NP-hard problem PC-TSP (see more details in Appendix A).

We target producing the “as-continuous-as-possible” (ACAP) toolpath for surface models, such that the number of transfer moves is minimized. Our key idea is to propose a “one-path-patch” (OPP) criterion to represent a surface patch that can be printed in a continuous toolpath, in combined flat and curved layers. We propose a bottom-up OPP merging algorithm for decomposing the given shell model into a minimal number of OPPs and generate the ACAP toolpath. This paper makes the following contributions:

- we introduce an original, fabrication constraints-aware, criterion “one-path-patch” (OPP) for representing a shell surface patch that can be printed in one path in the context of both flat layer and curved layer printing scheme.
- we propose a novel algorithm for decomposing the given shell surface into a minimal number of OPPs and generating the “as-continuous-as-possible” (ACAP) collision-free toolpath.
- we adapt our technique as a general computation framework for printing shell models in a “as-continuous-as-possible” manner on 3-axis extrusion-based printing platforms.

2 RELATED WORK

Slicing and Path Planning. In AM, the slicer is used for converting a model to toolpaths. Existing slicing algorithms divide the model as a stack of flat layers by geometric operations. [Lensgraf and Mettu 2016, 2017, 2018; Yoo et al. 2020] proposed a series of optimization algorithms to minimize the total extrusionless travel distance (wasted motion or print time) in the space of feasible toolpaths. We represent the precedence constraints as the dependency graph in a similar manner. In contrast, we define the OPP criterion, reform the optimization in a more compact dependency graph, and gain a more efficient optimization framework. For most cases, we can achieve the optimal results rather than approximated ones. Many efforts have been made to optimize the toolpath in terms of continuity, filling rates, and mechanical properties [Xia et al. 2020; Zhai and Chen 2019; Zhao et al. 2016]. Adaptive bead width control could further reduce the under- and over-filling artifacts [Hornus et al. 2020; Kuipers et al. 2020]. Hergel et al. [2019] present a method for generating strictly-continuous, self-supporting deposition paths for extrusion-based ceramic printing, performing nicely on watertight geometric models. However, this method cannot directly apply to shell models as illustrated in Figure 2.

Curved Layers Printing. Compared with traditional flat layers, curved layers contain dynamic z-values within individual layers.
and have excellent properties in AM. Such as alleviating the stair-case defect, improving surface smoothness, strengthening the printing model and reducing printing time. The first discussion about curved layers is called curved layers fused deposition modeling (CLFDM) [Chakraborty et al. 2008], which allows continuous change of the z-value within individual layers. Afterwards, [Allen and Trask 2015; B.Huang and S.Singamneni 2012; Llewellyn-Jones et al. 2016] perform physical experiments with FDM printers to demonstrate these properties. The industry standard slicing software Ultimaker Cura [Ultimaker 2021] also involves the curved layer printing in the surface mode that produces spiralized outer contours of the mesh instead of the solid model, which works well on simple shapes like a vase or cylinder. [Ezair et al. 2018] present an algorithm that generates covering curves based on geometric characteristics of a given volume. [Etienne et al. 2019] take a different approach that optimizes a parameterization to obtain smooth surface tops. The produced toolpaths are mapped back into the initial domain without requiring splitting or re-ordering. We apply this method in our curved OPP merging. Recently, researchers have applied curved layers printing on multi-axis printers. With the help of additional DOF, [Dai et al. 2018; Li et al. 2021; Xu et al. 2019] design curved toolpath to fabricate solid models in a support-free way. [Chen et al. 2019a] present a new CLFDM slicing algorithm that allows variable thickness layers. [Fang et al. 2020] introduce a field-based optimization framework to generate curved layers for reinforcing the mechanical strength of 3D printed models. However, we focus on three-axis printer platforms in this paper.

Fabrication of Thin Shells. Fabricating thin shells is gaining increasing attention as it accelerates the fabrication time compared to closed models and lightweight shell models are of wide applications. This advantage is further enhanced when fabricating viscous slurry materials, like clay and concrete, with large extrusion amount and fast deposition rate. For shell models, the continuity of material deposition is more critical due to the apparent artifacts caused by transfer moves. Most works in the literature adopt multi-axis platforms and incorporate curved layers for printing shells. [Mitropoulou et al. 2020] present a method to design non-planar layered print paths for robotic FDM printing of single-shell surfaces. [Bhatt et al. 2020] propose the layer slicing and toolpath planning algorithm to build thin shell parts on a 3-DOF build-platform and a 3-DOF extrusion tool. Printing concrete shells is attracting interests in the interdisciplinary area of digital fabrication and architecture. [Burger et al. 2020] use the single-shell as molds for concrete casting. [Anton et al. 2019] propose a design tool for producing bespoke concrete columns and involve curved layer for continuous extrusion. [Bhooshan et al. 2020] also emphasize on interactively shell modeling and integrate modeling with toolpath generation.

Decomposition for Fabrication. Many efforts have been focused on model decomposition for fabrication. Objectives for model decomposition include fabricating the model that satisfy the constraint, improving surface quality, saving or avoiding support structures, and reducing printing time. [Luo et al. 2012] propose a solution to decompose the model into smaller parts that every part can fit the printing platform. In addition to considering the criterion of printing volume, structural soundness and aesthetics are their decomposition objectives. For improving surface quality, [Hildebrand et al. 2013] generate a partition and compute the optimal slicing composition objectives. For improving surface quality, [Hildebrand et al. 2013] generate a partition and compute the optimal slicing direction for subparts. [Hu et al. 2014] decompose a given shape into as few approximate pyramidal parts as possible. Their motivation is that pyramidal shapes are well suitable for fabrication. [Vaneck et al. 2014; Wei et al. 2018] decompose shell models into small parts to save the support material and reduce the printing time. A manual assembly process is required after printing all shells. Also, to avoid supporting materials, [Wu et al. 2017, 2020]’s decomposition approaches consider the collision-free constraint and sequence of printing. They printed models in a multi-DOF 3D printing system so that manual assembly is not required. [Herholz et al. 2015; Muntoni et al. 2018] decompose general three-dimensional geometries to satisfy the height field constraint. To minimize the number of cutter setups for finish-stage machining in CNC, [Zhao et al. 2018] develop an algorithm to perform surface decomposition with the accessibility constraint. The above methods did not consider the continuity.
of the printing path as a criterion for model decomposition. It is worth mentioning that [Mahdavi-Amiri et al. 2020] propose carvability criteria for continually carving a connected domain, which requires both visibility and monotonicity. However, they did not take curved slicing layers into account. In such situation, the key difference between our OPP criterion and carvability is that OPP does not require visibility but support curved layering fabrication.

Ceramic Printing. Ceramic materials in AM have attracted heightened attention in recent years [Chen et al. 2019; Zocca et al. 2015]. It would reduce both processes and resources required to produce geometrically complex shapes in the traditional ceramics industry, and thus nurture new ideas or applications in architectural decorations [Chan et al. 2020], arts, etc. Researchers are also developing advanced engineering ceramics, such as metal oxides, carbides, and nitrides, to specific engineering demands [Peng et al. 2018]. Existing work mainly focuses on studying formulations of the water-to-clay ratios and some additives, and physical analysis of the sintered models in terms of compression, thermal stability, etc. [Ordoñez et al. 2019; Revelo and Colorado 2018].

Even though the slicing and toolpath planning for DII ceramic printing share both constraints and objectives with FDM, it possesses additional constraints due to the viscosity of clay, which is attracting the attention of researchers. Recent attempts consider the path planning for closed model [Hergel et al. 2019], integrated modeling and path generation for relatively simple shapes [Zhong et al. 2020], and stability enhancement for shell models [Xing et al. 2021]. While for general shell models, there are still no effective path planning methods.

3 OVERVIEW
Given a thin shell model $M$ with a feasible orientation that meets the support structure constraint, our algorithm aims to achieve maximal path continuity, i.e., decompose $M$ into a minimum number of surface patches where each patch can be printed consecutively. For narrative convenience, in this paper we define a printable surface patch as a path that can be printed with a single path.

As a PC-MPC problem, such decomposition is NP-hard that no efficient solution on all possible inputs. Our key idea is to apply an over-segmentation followed by a bottom-up merging procedure. We first slice $M$ through uniformly distributed flat planers (subsection 4.1). Each sliced element can be seen as a single printable surface patch. Multiple printing paths from mutually contiguous sliced elements can be connected into a single path by including a set of short connecting paths, which indicates that we can reduce the number of printable patches via merging small initial patches (subsection 4.2). Besides, we observe that curved slicing layers can be exceptionally effective in generating continuous printing paths for multiple separate printable surface patches of flat layers. The number of printable patches can be further reduced by replacing flat layers with curved slicing layers as much as possible (subsection 4.3).

The merging criterion is the main challenge for the bottom-up merging and curved layers replacement process. We need to formulate the "sub-surface-patches" that can be merged into a single printable patch or replaced by curved layers. The new concept of
We make an interesting observation that the possible collision differently when going uphill and downhill. See Figure 4, when going uphill (left), since the point on the right side of the outlet (green dot) is closest to the layer printed below, it may collide first, which can be avoided by restricting the path slope angle as $\theta_{\text{outlet}}$. Conversely, when going downhill (right), the nozzle has to be raised by $t$ to avoid self-layer collision between the point on the right side of the outlet (green dot) and the current layer. There is no need to define any slope angle for the downhill case with the raising operation. Finally, set the upper bound of the path slope angle as $\theta_{\text{max}} = \min(\theta_{\text{nozzle}}, \theta_{\text{object}}, \theta_{\text{outlet}})$. Note that this is a local constraint. The global collision caused by the height of the printed model exceeding the nozzle length is not considered. The solution will be mentioned in section 5.

**Support structure constraint.** We involve restrictions on the support structures for surface models, as support structures degrade the surface quality [Hergel et al. 2019]. For surface models that are not self-supporting, we decouple the fabrication of the intact model and support structures, i.e., we pre-print the support structures and place them during the fabrication process. This requires that the support structures locate on the ground and form a height field volume related to the printing orientation, as the support structures placed on the model may induce too much weight for the printed shell to afford. For such models, we decompose them into multiple patches and assemble them after fabrication (an example is shown in Figure 21).

### 3.2 One-Path-Patch (OPP)

Recall that we aim to maximize the continuity by decomposing the input shell model into the minimal number of printable "one-path-patches" (OPPs). With respect to a printing direction, a manifold surface patch is a printable OPP, if 1) there exists a set of slicing layers where each layer orthogonal to the printing direction intersects the patch resulting in a single *segment* or *contour*; 2) the resulting intersected *segments*/*contours* satisfies the fabrication constraints.

In the paper, we deliberately choose a low-resolution layer height to make the printing paths more visible. Three types of OPP can be defined according to the slicing layers: (I) only flat layers; (II) only curved layers; (III) combination of I and II, named as I-OPP, II-OPP, and III-OPP for short, shown in Figure 5. Curved layers of OPPs would take the thickness constraint and slope angle constraint.

What geometric properties should an OPP hold? For type I, the OPP criterion is equivalent to its *monotonicity*\(^1\), as (1,4) shown in Figure 5. As for type II, the intrinsic geometric properties of an OPP is hard to propose, where *monotonicity* becomes a sufficient and unnecessary condition. In Figure 5, (3,5,7) is an OPP but not monotone, and it demonstrates that a single OPP with curved layers could cover regions where multiple I-OPPs are applied.

To determine whether $P$ is an II-OPP, one option can be to extract curved layers directly and then assess the two fabrication criteria. CurviSlicer seems a perfect match for this, which desires to flatten as many areas as possible to minimize staircases [Etienne et al. 2019]. However, staircase minimization is not always equivalent to continuity maximization, which is sensitive to the target flat areas taken as the input of CurviSlicer, as shown in Figure 6. Two I-II-OPPs can be merged to a single II-OPP (details in subsection 3.3).

For a whole shell model, a bottom-up OPP merging procedure is introduced to minimize the number of OPPs, during which process III-OPPs (as 6,8,9 in Figure 5) are generated (subsection 3.1).

**Fig. 6.** An illustration that staircase minimization is not equivalent to continuity maximization. Left visualizes the CurviSlicer result with the target flat areas (initially attempt to flatten all areas under a specific slope angle, red segments). It could only flatten a part of the target flat areas to avoid violating layer thickness constraint. The result layers cannot produce OPP layers. Right shows the OPP layers and the specific target flat areas of our method. Note that different from CurviSlicer, the downward facing areas can also be taken as target flat areas thanks to the support structure constraint (subsection 3.1).

---

\(^1\) A 2D polygon $P$ is monotone with respect to a straight line $L$, if every line orthogonal to $L$, intersects $P$ at most twice. A 3D manifold surface patch is monotone in direction $L$ if all cross-sections orthogonal to $L$ are single section [Toussaint 1985].
we generate C’s top target flat areas by combining 1) OPP A’s top with our modified CurviSlicer satisfying the fabrication constraints. The red and blue segments indicate the top and bottom target flat areas (A₁, A₂, B₁, B₂). The green and purple lines indicate two oblique polylines.

be applied. Curved layers always produce fewer staircases associated with higher priority than flat layers.

(3) Two OPPs can be merged to one single OPP. Two OPPs (A and B) can be merged via two operations, stacking and curving, as shown in Figure 7. Stacking indicates that 1) OPP A’s bottom layer A₂ locates on the above neighboring layer of B’s top layer B₁; 2) A₂ and B₁ can be connected by its two end points. Curving indicates the merging operation of subsection 3.3. The two operations are used in subsection 4.3 to minimize the number of OPPs.

3.3 Curving Operation

The input of this operation are two I/II-OPPs (A and B) that can be originally merged via stacking. Curving operation aims to output the merged OPP (C) with its slicing layers. Suppose OPP A is above over B, as shown in Figure 7. Note that curving cannot be applied for two OPPs with only closed contours.

Our basic idea is to produce a modified version of CurviSlicer to extract curved layers for the merged OPP C. There are two key questions: 1) how to determine the top/bottom target flat areas of C? 2) how to guarantee the fabrication constraints? We solve the first question by combining the top/bottom target flat areas of A and B. For the determined top/bottom target flat areas of C, detect the slope angle constraint. Then extract the in-between curved layers with our modified CurviSlicer satisfying the fabrication constraints.

For an OPP, define the projection of its top (bottom) layer as its top (bottom) target flat areas (A₁, A₂, B₁, B₂), as shown in Figure 7. If both oblique lines violate the slope angle constraint, the two OPPs cannot be merged via curved layers. If not, we generate C’s top target flat areas by combining 1) OPP A’s top target flat areas (A₁), 2) the difference result of B’s top target flat areas and A’s bottom target flat areas (B₁ – A₂), 3) the oblique lines that meet the slope constraint; C’s bottom target flat areas are taken from B’s bottom target flat areas.

Next, we call CurviSlicer, specifying both top and bottom target flat areas. CurviSlicer formulated two key terms in their objective: one flat term to determine whether the target area can be flattened with the slope angle and thickness constraints and one smooth term to make the generated curved layer smoother. CurviSlicer with the smooth term would be much more time-consuming, which is unnecessary for our case, since the curving operation would be called frequently, and the generated layers would not be used in the final toolpath generation. So we only use the flat term while applying CurviSlicer. The merge is executable if the top and bottom target flat areas are successfully flattened without violating the fabrication constraints. Note that CurviSlicer only works for watertight 3D models. For the surface model, we convert it to an approximate watertight model with a minimal shell thickness that can be taken as the input of CurviSlicer (details in Appendix C).

4 ACAP METHODOLOGY

This section describes our algorithm in more details. For clear exposition, we explain the methodology for open 2D patches with only segments in each layer. The extension to 3D will be discussed in section 6. As introduced in section 3, the basic idea of our algorithm is a bottom-up OPP merging process based on a unified graph-based representation of the OPP graph and a set of graph nodes merging operations for the OPP graph. The OPP graph encodes the surface decomposition and their dependencies during the bottom-up OPP merging process. The OPP node merging operations are formulated through flat and curved slicing layers.

4.1 Building Dependency Graph

With an orientation that meets the support structure constraint (see Appendix B for details), we uniformly slice the model with flat planar layers vertical to the printing direction by the layer thickness (1.0mm in ceramic printing, 0.2mm in FDM), then build a directed acyclic graph to describe the dependency relationships, named dependency OPP graph Gdepend, where each node represents a sliced element (segment), each directed edge represents a dependency relationship between two neighboring nodes where the closest horizontal distance between them is smaller than the path width (6.0mm in ceramic printing, 1.5mm in FDM), shown in Figure 8(a). If node N₁ has a directed edge pointing to N₂, it indicates that (1) N₁ can only be printed after N₂, and (2) the two OPP nodes can be merged through stacking operation. A node can be printed only if all nodes it depends on have been printed.

4.2 OPP Merging through Flat Layers

This section aims at maximal continuity provided by flat slicing, that is, merging the flat sliced elements of Gdepend to a minimal number of I-OPPs. Each sliced element can be seen as an I-OPP, which can be merged by stacking operation (subsection 3.2). The merged I-OPPs should keep the dependency relationships formulated in Gdepend. Such merging process is indeed to find a path cover for Gdepend with the fewest paths considering the dependency relationships.

We propose two key steps for the merging process: 1) merge the nodes of Gdepend that must appear on the same path in any minimal path cover in advance, then build an initial OPP graph Ginit to reduce the size of Gdepend. 2) merge the nodes of Ginit further by solving the path cover problem with the dependency constraints. Rather than running an approximation algorithm, we propose a
we traverse all nodes of \( G \). As shown in Figure 8(c), the nodes are significantly reduced. Then compute the connected components. Each component acts as a node and edges are inherited from \( G_{\text{depend}} \).

![Fig. 8. (a): Dependency graph \((G_{\text{depend}})\), each node represents a sliced element, and each edge represents a dependency relationship between two nodes. (b): Temporarily ignore edges pointing to (from) nodes with in-degree (out-degree) not less than 2 (crossed by dashed red lines), and compute the connected components. (c): Initial OPP graph \((G_{\text{init}})\), each connected component of (b) acts as a node and edges are inherited from \( G_{\text{depend}} \).](image)

![Fig. 9. (a): Based on a path cover solution, the nodes of \( G_{\text{init}} \) are merged. (b): The path cover solution space \((G_{\text{solution}})\), where nodes (green dots) encode paths (merged I-OPPs) of path cover solutions, directed edges encode the printing order of the merged I-OPPs, and each path cover solution corresponds to a sequence of directed edges and nodes. Different sequences of \( G_{\text{solution}} \) may produce the same path cover solution but in a different printing order (two blue boxes). (c): Three sequences (two blue and one orange boxes) produce two optimal path cover solutions associated with the minimal number of \( G_{\text{flat}} \).](image)

**Path Cover Solution Space Exploration.** To merge \( G_{\text{init}} \) into the minimal number of I-OPPs, we aim to search for the shortest sequences while exploring \( G_{\text{solution}} \). We propose two key techniques to speed up such exploration by pruning the solution space. First, starting from the root node of \( G_{\text{solution}} \), we apply a beam search procedure with the branch and bound technique, to explore \( G_{\text{solution}} \) level by level, where the beam search width is set to \( W = 10^3 \) in our implementation. For each level of \( G_{\text{solution}} \), we sort its candidate nodes by the number of included \( G_{\text{init}} \)’s nodes, which implies that we tend to pick the nodes of \( G_{\text{solution}} \) including \( G_{\text{init}} \)’s nodes as many as possible. Note that nodes of \( G_{\text{solution}} \) would point to the same node of next level if its sequence included \( G_{\text{init}} \)’s nodes remain the same, such as \{\(1,2\), (4)\} and \{\(1,4\), (2)\} both point to (3,6) and (3,5). Second, for generating candidate nodes of each beam search iteration, we use a greedy strategy in the path cover solution generation with DFS, that each traversal would explore as deep as possible, as the traversal (1,3) will not terminate at node (3) since node (6) can be added into (1,3,6) shown in Figure 8(c). We have proved the optimality of this strategy\(^3\). The proposed path cover exploration method would produce multiple optimal \( G_{\text{flat}} \) with the least number of OPP nodes shown in Figure 9(c). The pseudo-code is presented in Appendix E.

4.3 OPP Merging through Curved Layers

Up to now, we obtained a set of unique \( G_{\text{flat}} \) with the maximal continuity via flat slicing. Could we further merge its OPP nodes? Recall that we observe curved slicing layers can be exceptionally effective in generating continuous printing paths for multiple separate printable surface patches of flat layers. Driven by this insight, we apply the curving operation (defined in subsection 3.3) as much as possible to further reduce number of OPPs. This can be done in two steps. First, apply the curving operation to sub-OPPs of each OPP node

\(^3\)This can be proved using proof by contradiction. If two adjacent sub-nodes belong to the two paths of an optimal solution, they must be terminal nodes of the paths. Obviously, the two paths can be further connected, which shows that the current path cover solution is not optimal.

\(^4\)In the traversal path process, if terminating the path when a node can be added, the node must be the starting point of another path. In the same way as the last proof, the number of paths of path cover, in this case, is at least 1 more than our strategy, so it is not optimal.
Another criterion is that the curving with three nodes (A). First, in the initial merging process, we merge the two OPPs (sub-OPP) nodes. The two merging processes are a deadlock of dependency after merging, shown in Figure 10(right). Since the three sub-OPPs of (c) accept stacking and only the three nodes connected by red edges in (a) are selected to merge. Since the three sub-OPPs of (c) accept stacking into one OPP, node 1 and 2 can be successfully merged. (Initial Merging Process), which would enlarge the target flat areas to be beneficial to the subsequent OPPs merging process. Second, apply the curving operation to merge multiple OPPs (OPP Merging Process). Below we propose a general pairwise-based merging procedure for both initial and OPP merging processes. We set \(G_{\text{flat}}\) as an initial curved OPP graph \(G_{\text{curved}}\), iteratively merge \(G_{\text{curved}}\) to implement the two merging processes. II-OPPs and III-OPPs would be generated accordingly.

**General Pairwise Merging Procedure.** OPP graphs are directed acyclic graphs (DAG). We propose an iteratively pair-wise merging procedure for a general DAG. For each iteration, randomly select an edge and try to merge the two related nodes with specific merging criteria to check whether they can be merged or not. If yes, update the graph by 1) erasing the edges of the two nodes, 2) merging a pair of nodes to one node and 3) connecting other related edges of the two nodes to the merged node. The terminal condition is that no pairs of nodes can be merged.

**Merging Criteria of OPP Graph.** In our case, the directed edges of OPP graph indicate dependency relationships of OPPs. While merging a pair of nodes, one necessary criterion is to guarantee there are no multiple paths between the two nodes, since it will result in a deadlock of dependency after merging, shown in Figure 10(right). Another criterion is that the curving operation can be applied to the two OPPs (sub-OPP) nodes. The two merging processes are demonstrated in Figure 11(A~B) and (B~D). For each iteration of OPP Merging Process, we aim to merge two OPP nodes with the curving operation, which is taken as an inner loop of the OPP Merging Process (see a~c in Figure 11).

**Inner Loop of OPP Merging Process.** As for the inner loop, a DAG can be formulated from the two candidate merging OPP nodes: 1) take their sequences of sub-OPPs where each sub-OPP is a node and maintain the dependency edges of these sub-OPPs; 2) add back the associated dependency edges between the two sequences from \(G_{\text{init}}\). While applying the pairwise merging strategy to the result DAG (Figure 11(a~c)), we only select edges that benefit merging the two sequences to a single sequence. Specifically, we label the nodes that have edges across the two sub-OPP sequences (two red edges shown in (a)), then select edges over such labeled nodes. For each pair of nodes, call curving operation to merge the two nodes (subsection 3.3). The pseudo-code is presented in Appendix E.

**Optimality of Proposed Method.** For different \(G_{\text{curved}}\) of input, the final number of nodes after merging may be different. We randomly select one \(G_{\text{curved}}\) with the least number of nodes since we only consider the OPP number criterion. Different \(G_{\text{curved}}\) and order of merging OPPs by curving may result in curved layers with different distributions, in other words, different sub-OPP of the final III-OPP, as shown in the inset. Similarly, since the order of selecting nodes in two levels is random, we cannot guarantee the global optimal solution. Figure 23 shows an example, and more details are discussed in Appendix D.

**4.4 Layers Connection**

With the bottom-up merging OPP process, we get an optimal OPP decomposition. This section works on path planning for each OPP by converting OPP slicing layers to continuous toolpath and generating transfer moves between OPPs. Note that we remove the most time-consuming smooth term of CurviSlicer while applying the curving operation during OPP merging (subsection 3.3). Here we add it back and rebuild the smoother curved layers of related OPPs for the toolpath planning. Then connect inter-layer to a single path for each OPP and determine the fabrication order of these OPP paths.

**Inter-layer Connection Path.** Since there are only segments in 2D models, they can be connected using Zig-zag pattern. For the two terminal points of a segment, if one is the entry point, the other will be the exit point. We select entry points for all segments, minimize the total length of the connection path between the entry and exit points of adjacent segments. Then we set a Euclidean distance threshold \(D\) to determine whether two terminal points of two neighboring layers can be connected directly with a straight segment. If the distance exceeds \(D\), an extra path is added. See the inset, call the current terminal point in the current layer \(P_{\text{source}}\), and the terminal point to be connected in the next layer \(P_{\text{target}}\). Print along the current printed layer with \(t_{\text{min}}\) layer thickness from \(P_{\text{source}}\) to the position closest to \(P_{\text{target}}\), and
The example in Figure 12(b) shows that three candidate collision dependency edges (purple) are supposed to insert in $G_{\text{depend}}$. (c): Delete two redundant candidate collision dependency edges. (d): Perform the method for all layers and finally three collision dependency edges are added to $G_{\text{depend}}$. The orange boxes show the merging results in $G_{\text{init}}$ without adding collision dependency edges.

then print along straight line to $P_{\text{target}}$ (the orange path in the inset). The post path optimization in section 6 will improve the spatial distribution of the generated extra path.

OPPs Sequence. With the known dependency relationships of these OPP paths, we apply a method as subsection 4.2 to produce a feasible fabrication order. Then, we plan travel moves between OPP paths by withdrawing the nozzle to a safe distance above the printed objects to avoid possible collisions (see Figure 1). Finally, a G-code file is generated to transfer the toolpath to the printer.

5 GLOBAL COLLISION CONSIDERATIONS

The algorithm of section 4 does not consider the global collision raised by the printed parts exceeding the nozzle length shown in Figure 12(a), which often occurs when printing a lower layer after the higher parts. To extend our method with this consideration, the key challenge is how to represent the global collision constraint in our proposed bottom-up OPP merging algorithm. We intend to formulate such constraints to a new type of directed edges of the OPP graph, named "collision dependency edges." Similar to the original directed edges, the new edges would represent printing dependencies. The difference is that the OPP nodes of the new edges cannot be merged through stacking operations. We would first clarify the generation of collision dependency edges and then introduce the algorithm's modifications raised by the new edges.

Generation of Collision Dependency Edges. To append the novel edges of $G_{\text{depend}}$, we apply the same method to model the printing nozzle in subsection 3.1. For each node pair of $G_{\text{depend}}$, add a collision dependency edge between them if a collision occurs while printing. As the layer "3" and "4" in Figure 12(a), the collision indicates that the layer "4" must be printed after the layer "3". Note that we maintain $G_{\text{depend}}$ as a Hasse diagram [Pemmaraju and Skiena 2003], where redundant dependency edges do not exist. For example, if exist two edges: $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$ is a redundant edge. The example in Figure 12(b) shows that three candidate collision dependency edges starting from the node "3" has been compressed into one single edge (c).

![Fig. 12. Illustration of collision detection and building dependency graph. For layer "3", traverse and perform collision detection with other layers, and collision occurs (a). (b): The layer "3" will collide with three layers, thus three candidate collision dependency edges (purple) are supposed to insert in $G_{\text{depend}}$. (c): Delete two redundant candidate collision dependency edges. (d): Perform the method for all layers and finally three collision dependency edges are added to $G_{\text{depend}}$. The orange boxes show the merging results in $G_{\text{init}}$ without adding collision dependency edges.](image)

![Fig. 13. Zig-zag connecting path for segments (left) and Spiral connecting path for contours (right).](image)

Algorithm Modifications. 1) For subsection 3.3, we need to add a requirement for applying curving operation, to guarantee no collision dependency between the top and bottom target flat areas of two OPPs. 2) For subsection 4.2, the method of path cover solution exploration remains the same as above. However, suppose a node of path cover solution space ($G_{\text{solution}}$) includes sub-OPP nodes with "collision dependency edges", these sub-OPP nodes cannot be merged in $G_{\text{flat}}$. 3) For subsection 4.3, we observe that the number of OPP nodes (6 nodes) in $G_{\text{init}}$ becomes much more than that without including collision dependency edges (3 nodes indicated with orange boxes), as shown in Figure 12(d). To solve the efficiency problem raised by the increasing of OPP nodes, especially for the OPP merging procedure with curved layers, we add an external step before the Initial Merging Process. That is to apply the stacking operation for sub-OPPs of each OPP node according to the $G_{\text{init}}$ which is generated without considering the global collisions.

6 EXTENSION TO 3D

Extending our algorithm from 2D to 3D does not require extra effort for most steps, except to deal with contour in the $G_{\text{init}}$ construction step and extend the toolpath generation step for the 3D case. In this section, we describe the extension of these steps in detail.

Initial OPP Graph. For 3D cases with contours, we first build a $G_{\text{init}}$ with the method of subsection 4.2. If an OPP node of $G_{\text{init}}$ has both segments and contours, divide it into pure segment nodes and pure contour nodes. Such classification is conducive to the next step of merging via curving, which only allows the input of two OPPs composed of segments, or one OPP composed entirely of segments, and the other entirely of contours.

Contour Spiralization. To spiralize contours $(C_1, ..., C_n)$, we need first to determine one connecting point for each contour. As subsection 4.4, we aim to minimize the total distance between connecting points of adjacent contours. We design a dynamic programming algorithm to choose appropriate connecting points for each contour. We discretize each contour with $m$ sampling points, where $P_{ij}$ is a sampling point in $C_i$. Denote $d_{ij}$ as the minimum sum of length between neighboring contours $C_1, ..., C_i$ when choosing $P_{ij}$ as connecting point. The transition equation is as below:

$$
d_{ij} = \min_{k=1, ..., m} (d_{(i-1)k} + \|P_{(i-1)k} - P_{ij}\|), \quad i \neq 1$$
Then, we connect these contours by spiral path, shown in Figure 13. For each two adjacent contours, interpolate all sampling points starting from the connecting points:

\[ P_i = w \cdot P_a + (1 - w) \cdot P_b, \quad w = \frac{S_{\text{current}}}{S_{\text{total}}} \]

where \( P_b \) is a sampling point of the below contour (if \( S_{\text{current}} = 0 \)), \( P_b \) is connecting point of current contour, \( P_a \) is the nearest point to \( P_b \) of the above contour, \( S_{\text{total}} \) is total length of the below contour, \( S_{\text{current}} \) is the geodesic distance that \( P_b \) moves from the connecting point along the below contour, \( P_i \) is the interpolation point. Note that the top contour as a boundary will not be spiralized.

**Filling the Low-Slope Area.** While slicing, inadequate layers over low-slope regions inevitably result in under-fills, as shown in Figure 14(left). Such under-fills of small areas can be removed by a post path optimization process (see Figure 15 left bottom). For large under-fill areas, we propose to fill them with connected Fermat spirals [Zhao et al. 2016]. In our implementation, we set a threshold of 20’ to detect the Fermat spiral region. We traverse each pair of the adjacent contours, generate the matching edges between the sampling points of two contours by minimal Euclidean distance, and measure the angle of these matching edges from the horizontal plane. If all angles of these matching edges are smaller than the threshold, add the surface path between the two contours to the Fermat spiral region. Figure 14(right) shows the filling path over the original 3D surface.

**Toolpath Optimization.** Since we generate toolpath for each OPP separately, the adjacent paths may be too close or too far from each other (left center of Figure 15(left)). Some inter-layer connection paths (subsection 4.4) and the filling paths of low slope areas (left bottom of Figure 15(left)) may be not uniformly distributed (left top of Figure 15(left)). To eliminate these problems, we use a similar method as [Zhao et al. 2018] to optimize the final toolpath, in which they iteratively evolve a single toolpath considering the spacing and smoothing constraints. Different from the original method, our input is not a single path but multiple continuous paths. Figure 15 shows a result before and after optimization.

7 RESULTS AND DISCUSSIONS

This section shows OPP decomposition results on surface models with varying geometric complexity, printed results, and comparisons with the Ultimaker Cura 4.9.1 software. We evaluate our ACAP algorithm on DIW-based ceramic printing and FDM platforms, with clay and thermoplastics as the material, respectively.

7.1 Implementation and Parameters

Our algorithm is implemented with C++, running on a PC with an Intel Core i7-9700 CPU @ 3.0GHz and 32GB memory. For the printing experiments, we use a 3-axis DIW ceramic printer Eazao Mega 5 with 470×370×390 mm³ printing volume (Figure 16) and an FDM printer Hori Z560 with 360×350×500 mm³ printing volume. Note that the parameters below in the brackets refer to the FDM. For the ceramic printing (FDM) printer, we use a 90 (8)mm long nozzle with the nozzle diameter of 5.2 (1.0)mm (Figure 16 shows the ceramic printer’s nozzle), nozzle movement speed as 25.0 (25.0)mm/s. The printing path width is set to 6 (1.5)mm. In subsection 3.1, we take the average of the layer thickness range, i.e., \( t = \frac{1}{3} (0.35) \) mm for calculating slope angle constraint, which is 30° (35°) according to the formulation. We set the flat layer thickness to 1.0 (0.2)mm for slicing, producing the best surface quality in our experiments, and also use it as the slicing layer thickness to slice models and generate \( G_{\text{depend}} \). The distance threshold \( D \) is 5 (2)mm in connecting layers (subsection 4.4).

7.2 Fabrication Results

Figure 1, Figure 17 and Figure 18 show results of two different printing processes (ceramic printing and FDM), including OPP decomposition, printing path, and printed models (using our method.
As-Continuous-As-Possible Extrusion Fabrication of Surface Models

Fig. 17. Some ceramic 3D printing results. The models in the top three rows are self-supporting. The OPP decomposition, toolpath, and printed results by our method and Cura are shown for each row, respectively. The last three rows show models with support. We pre-build the support structures and insert them manually during fabrication.

and Cura, respectively. Refer to Table 1 for the statistics, where we list each model’s height, the number of OPPs produced by our algorithm, and the printing time raised by our method and Cura software. Specifically, the table includes the number of OPPs after merging through flat layers (#OF), and the number of OPPs after merging through curved layers (#OO), to illustrate the effectiveness of stacking and curving operations. Note that we scale models to half size in FDM experiments for time saving. Since the nozzle used for FDM printing is much shorter than for ceramic printing, our method partitions more OPPs to avoid the global collisions.

Compare with Cura. The top three models in Figure 17 and the four models in Figure 18 are all self-supporting. In the setting of Cura, we choose the Surface Mode, enable Spiralize Outer Contour and Retraction, and use the same nozzle movement speed as our method. It shows that our method outperforms Cura for shell models containing multiple contours or segments in both surface quality (fewer artifacts and slighter deformation) and fabrication efficiency (save 24%–48% of printing time). For example, as the Grail model printed with Cura (ceramic printing), a large number of transfer moves (1541) induce deformation (see the red circle in Figure 17). Such transfer moves would increase the forces on the printed part and may cause the collapse (see the yellow circle). In contrast, the ACAP toolpath for this model only has one transfer move in FDM, and even no transfer move in ceramic printing, and thus strengthens the model during fabrication and avoids these problems. Generally, our algorithm improves the printing efficiency more significantly for the models with multiple branches like the Julia vase or TPMS model. Moreover, we can see the benefit of curved layers, which reduces the number of OPPs by around 50%, and improves the staircase defect (see the upper part of the Crown model). The TPMS model has no curved layer, as the slope angle constraint restricts it.

For the models that require support structures, we pre-build the support structures and place them before printing the models. See the last three models in Figure 17. To make support structure easy to remove, we add a membrane on the surface of their contact before printing the model.

Discussions on the printing quality. Generally, the quality of ceramic printing is more sensitive to toolpath continuity for surface models than FDM printing. Therefore, the clay results from the ACAP algorithm show significant superiority in the model quality over those from Cura. However, we can still observe some visible
artifacts in the clay printouts, like over-extrusion at the zig-zag turning areas, seams at the junction of OPPs, and sagging at the concavity corner. The primary reason is that the DIW-based ceramic printing technique with clay is less mature than FDM with thermoplastics. The extrusion amount can hardly be precisely controlled due to material inertia. Moreover, there are many uncertainties during the fabrication process, such as material humidity, air pressure, and other coupled hardware and material problems.

We also remark a defect in curved layers. See Figure 19; we observe that the layer thickness is non-uniform in the outer boundary, appearing like the stair-case defect. The reason is that curved slicing layers are applied with adaptive layer thickness, and the material deforms in different behaviors when extruded in downhill or uphill directions. Such artifact might be improved via fine-tuning the material flow or the nozzle height, while highly demanding on three issues: stable properties of the clay material, accurate simulation of clay deformation, and precise control of the extrusion rate for clay in the printing platform.

7.3 Algorithm Performance

Effect of nozzle length. The nozzle size, especially the nozzle length, also affects the number of OPPs, as the shorter nozzle indicates more possibilities for the collision between the model and the printing platform. See Figure 20, we test the nozzle length from 5mm to 30mm on the Julia vase model in three different sizes.

Running time. Table 2 shows the running time statistics with the number of nodes and edges of the OPP graph during each step of our ACAP algorithm. For most of the input models, our algorithm is extremely efficient in the generation of $c_{\text{depend}}$, $G_{\text{init}}$ and $G_{\text{flat}}$, since our algorithm does not involve any geometric computation in this phase, but only some graph operations. The only exception is the TPMS model (FDM), whose time of building $G_{\text{depend}}$ and $G_{\text{flat}}$ running time for merging $G_{\text{curved}}$ ($T_d$), performing CurviSlicer with smooth term and the layers connection ($T_c$), finishing toolpath optimization ($T_o$), and the total running time ($T_t$).

![Figure 20](image-url)
As-Continuous-As-Possible Extrusion Fabrication of Surface Models

8 CONCLUSION, LIMITATION, AND FUTURE WORK

In extrusion-based 3D printing, path continuity significantly impacts the surface quality and printing time, especially for shell models. We put forward the original concept of OPP to quantify path continuity and propose a method to decompose the given shell model into as few OPPs as possible, considering manufacturing constraints on a standard three-axis printer platform. We demonstrated our methods on various models, and the results are superior to existing methods in surface finish and printing time.

Limitations and Future work. Regarding the toolpath, improvement space still exists. The inter-layer connecting path might be outside the model by a distance \( t_{\text{min}} \), and such an artifact cannot be guaranteed to be avoided through the post global toolpath optimization. The constant path width maintains the uniform horizontal wall thickness of the shell model; however, the shell thickness in the direction normal to the surface varies, which is an unintended consequence.

Our framework only allows for geometry to be printed which is surface model. Extending the OPP criterion and the ACAP algorithm to general solid models could be valuable continuous work. The key issue is how to adapt the OPP criterion to different interior structures or infilling patterns. Moreover, to fully explore the applications of surface models, applying the shell reinforcement techniques like adding ribs [Gil-Ureta et al. 2020] or modulating the thickness [Xing et al. 2021] is also an exciting direction. In such a way, the contour of the surface models is of varying thickness. The ACAP algorithm should be adapted to combine both thin regions of +/- the nozzle size and wider ranges of multiple nozzle sizes.

The support structure constraint limits the feasible surface shapes for fabrication, i.e., the model is either self-supporting or with the support structures located on the ground. Thus, for models with no orientation satisfying the support structure constraint, we decompose the model manually and assemble them afterwards, as shown in Figure 21. We remark that different degrees of shrinkage of material may leave noticeable seam lines between parts after solidifying of material. In future work, we would like to study the generalization of more complex surface models by considering more decoupling strategies between the support and intact structures, and also the scheduling strategies for printing the support structures in-situ with the intact model. An automated pre-decomposition that balances the number of decomposed patches and path continuity is also a natural objective.

Since our method generates the toolpath and related G-code files, it would be natural to cooperate with the research works that fine-tune the printing parameters like the extruder height and extrusion amount [Takahashi and Miyashita 2017], or positions in a local range [Yan et al. 2021], to achieve various delicate geometric features without violating the toolpath continuity. Referring to Figure 22, each OPP can be treated as an independent unit to embed different textures on the fly.

Finally, another problem worth exploring is extending the algorithm to multi-axis printing setups. Accordingly, the slope angle constraint can be ignored, the support structures are of fewer restrictions, and the toolpath has more freedom. We believe the research direction owns great potential.

REFERENCES

Robert J.A. Allen and Richard S. Trask. 2015. An experimental demonstration of effective Curved Layer Fused Filament Fabrication utilising a parallel deposition robot. Additive Manufacturing 8 (oct 2015), 78–87. https://doi.org/10.1016/j.addma.2015.09.001

Ana Anton, Angela Yoo, Patrick Bedarf, Lex Reiter, Timothy Wangler, and Benjamin Dillenburger. 2019. Vertical Modulations. Computational design for concrete 3D printed columns. In ACADIA 19: Ubiquity and Autonomy. Proceedings of the 39th Annual Conference of the Association for Computer Aided Design in Architecture, Kory Bieg, Danelle Briscoe, and Clay Odom (Eds.). Association for Computer Aided Design in Architecture (ACADIA), s.l., 596 – 605.

Prahar M. Bhatt, Rishi K. Malhan, Pradeep Rajendran, and Satyandra K. Gupta. 2020. Building free-form thin shell parts using supportless extrusion-based additive manufacturing. Additive Manufacturing 32 (mar 2020), 101003. https://doi.org/10.1016/j.addma.2019.101003

Shajay Bhooshan, Tom Van Mele, and Philippe Block. 2020. Morph & Slerp: Shape description for 3D printing of concrete. Symposium on Computational Fabrication (nov 2020). https://doi.org/10.1145/3424630.3425413
The precedence constrained traveling salesman problem (PC-TSP) was formulated by [Kubo and Kasugai 1991] and proved to be NP-hard: "Given a directed complete graph \( G(V, E) \), a distance \( D_{ij} \) on each arc \((i, j) ∈ E\), precedence constraints \( \prec \) on \( V \), find a minimum distance tour that starts node 1 ∈ \( V \), visits all the nodes in \( V - \{1\} \), and returns node 1 again so that node \( i \) is visited before node \( j \) when \( i \prec j \)." We define our precedence constrained minimum path cover problem (PC-MPC): given a directed acyclic graph \( G(V, E) \) where edges are precedence constraints \( \prec \) on \( V \), our goal is to find a path cover for \( G \) with the fewest paths.

To prove that PC-MPC is NP-hard, we reduce this problem to PC-TSP for the input of PC-MPC \( G \) to turn it into a weighted directed complete graph: (1) set the weight of directed edges of \( G \) to 0. (2) add missing edges to make \( G \) a directed complete graph \( G_{com} \) where the weight of new edges is set to 1. Therefore, we convert the input of PC-MPC \( G, \prec \) to the input of PC-TSP \( G_{com}, D, \prec \) in linear time. The goals of PC-TSP and PC-MPC are equivalent by this conversion. With an arbitrary starting node meeting precedence constraints, find a tour from this node by solving PC-TSP. The output of PC-TSP can be taken as the output of PC-MPC by removing edges with a weight of 1 on tour. The minimum distance of PC-TSP plus equals the number of paths for the minimum path cover of PC-MPC.

Given the input model \( M \), this step extracts feasible printing orientations by uniformly sampling orientations over the Gaussian sphere, and then choosing one that satisfies the support structure constraint. For each extracted orientation, apply a similar method as [Hergel et al. 2019] to detect the support areas and validate the support structure constraint. First, slice the model and if a layer sampling point requires support, cast a ray downward. If intersect with the model itself (except adjacent layer), it indicates violating the support structure constraint.

CurviSlicer takes the watertight triangle mesh, its tetrahedral mesh, and target flat areas as input, which cannot be directly applied to the surface model, a surface patch of watertight triangle mesh. The key challenge is that it’s not straightforward to extend the gradient formulation of the vertical coordinates within each tetrahedron (Sec. 4.2 of [Étienne et al. 2019]) to that formulation within each triangle. We propose to generate a tetrahedral mesh to approximate the original surface patch of watertight triangle mesh with a minimal shell thickness. In our implementation, we set it to 0.3% of the longest diagonal of the bounding box of the input model. We do not suggest directly offsetting the input surface model along the horizontal direction to form a watertight mesh and generate the corresponding tetrahedral mesh. The self-intersection problem raised by offsetting makes it hard for tetrahedralization. Our solution is described below. For each triangle, offset its centroid by a minimal distance along the triangle’s normal direction, then connect the resulting point with the three points of the triangle to form a tetrahedron. For each edge of the target flat boundaries (a set of edges of triangles), offset its midpoint by a minimal distance along the horizontal direction starting, then connect the resulting point with the two endpoints to form a triangle.
Algorithm 1 OPP Merging through Flat Layers

1. **Input:** $G_{\text{init}}$: The beam search width $W$; 
2. **Output:** A set of optimal flat OPP graphs $\mathcal{O}$;  
3. Data structure setting; 
4. A directed acyclic graph $G_{\text{solution}}$ to encode the solution space; 
5. A nodes vector $P$ of the previous depth of beam search; 
6. A nodes vector $C$ of the current depth of beam search; 
7. Set $G_{\text{solution}}$’s root node to an empty virtual node; 
8. Add $G_{\text{solution}}$’s root node to $P$; 
9. while $P \neq \emptyset$ do 
   // Generate candidate nodes of current depth from $P$ 
   10. Declare a set of node sequences $S$; $S \leftarrow \emptyset$; $C \leftarrow \emptyset$; 
   // Inner Loop of OPP Merging Process 
   11. for each node $\diamondsuit$ of $P$ do 
      12. Get the node sequence $s_i$ from $G_{\text{solution}}$’s root node to $\diamondsuit$; 
      13. Get the corresponding flat OPP graph $G_{flat}$ of $s_i$; 
      14. Get the set $M_i$ of $G_{\text{init}}$ which are included in $s_i$; 
      15. if $M_i = $ all nodes of $G_{\text{init}}$ & $G_{flat}$ $\notin S$ then 
         16. $O \leftarrow O \cup \{G_{flat}\}$; 
         17. Continue; 
   // Generate candidate nodes from $\diamondsuit$ 
   18. Get $G_{\text{init}}$ from $G_{\text{init}}$ by deleting related edges to $M_i$; 
   19. for each node $\heartsuit$ of $G_{\text{init}}$ do 
      20. if $(\heartsuit$’s in-degree == 0 and $\heartsuit$ $\notin M_i$) then 
         21. Continue; 
      22. Explore all traversal paths $P_j$ starting from $\heartsuit$ with 
         the greedy strategy to explore $G_{\text{init}}$ as deep as possible; 
   // Update $G_{\text{solution}}$ and $C$ 
   23. for each traversal path $P_k \in P_j$ do 
      24. Take $P_k$ as a new candidate node $\clubsuit$ of $G_{\text{solution}}$; 
      25. $b \leftarrow$ whether a sequence exists in $S$ with the 
         same $G_{\text{init}}$’s nodes as ($s_i; \heartsuit$); 
      26. Declare the last node of selected sequence $\spadesuit$; 
      27. if $b \& (\heartsuit = \spadesuit)$ then 
         28. Add an edge from $\heartsuit$ to $\spadesuit$ in $G_{\text{solution}}$; 
      29. else 
         30. Add $\heartsuit$ and an edge from $\heartsuit$ to $\heartsuit$ in $G_{\text{solution}}$; 
      31. $C \leftarrow C \cup \{\heartsuit\}$; 
      32. $S \leftarrow S \cup \{(s_i; \heartsuit)\}$; 
   // Apply the branch and bound technique 
   33. if $O \neq \emptyset$ then 
      34. Return $O$; 
   // Extract $W$ nodes from candidate nodes of current depth 
   35. Sort $C$ by the number of included $G_{\text{init}}$’s nodes; 
   36. $P \leftarrow$ the first $W$ nodes of $C$; 

Algorithm 2 OPP Merging through Curved Layers

1. **Input:** A $G_{flat}; G_{\text{init}}$; 
2. **Output:** A $G_{curved}$; 
3. Two nodes $n_l, n_r$ of $G_{curved}$; 
4. $G_{curved} \leftarrow G_{flat}$; 
   // Initial merging process 
5. for each node $\diamondsuit$ of $G_{curved}$ do 
   6. Pairwisedly merge sub-OPPs of $\diamondsuit$ with curving operation; 
   // OPP merging process 
7. $b \leftarrow True$ 
8. while $b$ do 
   9. $b \leftarrow False$ 
10. for each edge $(n_l, n_r)$ of $G_{curved}$ do 
   11. // Deadlock detection 
      if more than one path between $n_l$ and $n_r$ then 
         12. Continue; 
   13. // Inner Loop of OPP Merging Process 
   14. Formulate a DAG $G_{sub}$ from sub-OPPs of $n_l$ and $n_r$; 
   15. Get two sub-OPP sequences of $n_l$ and $n_r$; 
   16. Add back the edges between two sequences in $G_{\text{init}}$; 
   17. Pairwisedly merge nodes of $G_{sub}$ with curving operation; 
   18. Label the nodes that have edges across the sequences; 
   19. Select edges of $G_{sub}$ over such labeled nodes; 
   20. for each selected edge, call curving operation; 
   // Update $G_{curved}$ 
21. if $G_{sub}$ has been merged to a single OPP node then 
   22. Update $G_{curved}$ by merging $n_l$ and $n_r$; 
   23. Update sub-OPP sequences of the merged OPP node; 
   24. $b \leftarrow True$; 
   25. Break;