# An ACO-Based Tool-Path Optimizer for 3D Printing Applications

Kai-Yin Fok, *Student Member*, *IEEE*, Chi-Tsun Cheng, *Member*, *IEEE*, Nuwan Ganganath, *Member*, *IEEE*, Herbert Ho-Ching Iu, *Senior Member*, *IEEE*, and Chi K. Tse, *Fellow*, *IEEE*

***Abstract*—Layered additive manufacturing, also known as 3D printing, has revolutionized transitional manufacturing processes. Fabrication of 3D models with complex structures is now feasible with 3D printing technologies. By performing careful tool–path optimization, the printing process can be speeded up, while the visual quality of printed objects can be improved simultaneously. The optimization process can be perceived as an undirected rural postman problem (URPP) with multiple constraints. In this paper, a tool–path optimizer is proposed, which further optimizes solutions generated from a slicer software to alleviate visual artifacts in 3D printing and shortens print time. The proposed optimizer is based on a modified ant colony optimization (ACO), which exploits unique properties in 3D printing. Experiment results verify that the proposed optimizer can deliver significant improvements in computational time, print time, and visual quality of printed objects over optimizers based on conventional URPP and ACO solvers.**

***Index Terms*—Layered additive manufacturing, arc-routing, rural postman problem, ant colony optimization, tool–path optimization**

## I. Introduction

IN a typical fused deposition modelling (FDM) based 3D printing process, a computer–aided design (CAD) file of a 3D model is fed into a slicer software for breaking it down into numerous thin layers. Each layer of the model will then be decomposed into print segments and further be converted into control codes for machining motions of the mechanical parts in a FDM machine. Printing nozzles of most off–the–shelf FDM machines move on the surface of their print beds, while their extruders control the flow of filament. Molten filaments, which are usually made of polylactic acid (PLA) or acrylonitrile butadiene styrene (ABS), will be deposited via the printing nozzle onto the print bed to construct the model layer by layer. To print a segment, the nozzle first moves to the start point of the segment and then traverses to its end. Meanwhile, the extruder injects filament toward the reservoir of the nozzle and creates the required pressure. Molten filament is then pushed out of the nozzle and forms the print segment. Before the nozzle reaches the end of the segment, the extruder reduces the pressure gradually such that no extra filament is deposited beyond the end of the current segment. The nozzle then moves to the start point of the next segment. The process repeats until all print segments on the current layer have been traversed. The print bed is then descended and the printing process of the next layer proceeds.

During a printing process, most of its print time is spent on moving the printing nozzle along print and non–print segments, which are also known as *transitions*. Transitions are movements of the printing nozzle among disjoint print segments. While the time spent on traversing print segments cannot be reduced as the length and velocity for the printing nozzle to traverse each print segments are predefined, the total print time can be shortened by having shorter transitions.

Apart from object print time, visual quality is also regarded as an important criteria in 3D printing. One of the major causes for a degradation in visual quality is the existence of *strings*. Strings are referring to the residual material that remains on the surface of a printed model. Since a 3D model is constructed layer by layer, even if the model has a single continuous structure in the 3D space, disconnected regions can be found on some of its layers depending on its shape and orientation while printing. Whenever the nozzle hops across boundaries of discrete regions, excess filament could leak from the nozzle unintentionally and form strings on the model. Typical 3D printers alleviate the strings issue by performing retraction, which is a relatively time consuming process as it creates the required suction at the nozzle by using its extruder to withdraw filament from the reservoir. Nevertheless, as observed in our experiment results in Fig. 1(c), such method alone is insufficient in tackling the strings issue.

Results shown in Fig. 1(c) were obtained using an ordinary domestic 3D printer [1] and a popular slicer software [2]. The printer is carefully calibrated to yield high structural accuracy and surface texture resolution (see. Fig. 1(b)). However, as shown in Fig. 1(c), traces of strings can still be observed even when retraction is enforced. Excessive strings appear between two disjoint parts on the same layer suggest there are rooms for further optimizations.

In this work, an ant colony optimization (ACO) based tool–path optimizer for 3D printing applications is proposed to shorten printing processes and improve visual quality of printed models simultaneously. The proposed optimizer accel-

Manuscript received July 10, 2017. Accepted for publication December 6, 2018. This work is jointly supported by the Department of Electronic and Information Engineering, Hong Kong Polytechnic University, the School of Electrical, Electronic and Computer Engineering, the University of Western Australia, and the Department of Manufacturing, Materials and Mechatronics, RMIT University.

Kai-Yin Fok and Chi K. Tse are with the Department of Electronic and Information Engineering, the Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong (email: kyfok@ieee.org; michael.tse@polyu.edu.hk).

Chi-Tsun Cheng is with the Department of Manufacturing, Materials and Mechatronics, RMIT University, 124 La Trobe St, Melbourne VIC 3000, Australia (email: ben.cheng@rmit.edu.au).

N. Ganganath and H.H.C. Iu are with the School of Electrical, Electronic and Computer Engineering, the University of Western Australia, Crawley, WA, Australia (email: nuwan@ganganath.lk; herbert.iu@uwa.edu.au).

erates printing processes by eliminating unnecessary movements of the printing nozzle and alleviates the strings issue in 3D printing by only allowing the nozzle to hop across boundaries of disjoint parts when necessary. Section II discusses some existing works which have been done on related topics. Section III provides the problem formulation. The proposed tool–path optimizer is introduced in Section IV. The problem solvers utilized in the proposed optimizer are introduced in Section V. A modified ACO optimizer is proposed in this paper, which exploits the unique characteristics in 3D printing. The modifications are elaborated in the same section. The proposed optimizer was evaluated using both simulations and actual 3D printing experiments. Results are presented, analyzed, and discussed in Section VI. Concluding remarks are given in Section VIII.

## II. Literature Review

Accuracy and surface finishing of printed objects have always been important criteria in additive manufacturing. In [4], Armillotta studied the surface quality of textured printed objects and provided guidelines on selecting suitable object scale and orientation without compromising the texture quality. Agarwala *et al.* [5] further considered the precision of internal structure in fabricating ceramic and metal printed components. Their physical properties were also studied. A similar study was conducted by Lim *et al.* [6] on concrete printing in construction applications. In contrast to subtractive manufacturing, interiors of 3D printed objects are normally filled with infilling segments. Jin *et al.* [7] tried to improve printing quality and accuracy by generating parallel infilling segments with adaptive separations. Recently, Freens *et al.* [8] proposed a method for optimizing the production planning of a 3D printing factory. They formulated the problem as an extension of a bin packing problem with lateness and special requirements from the printed models. Simulation results show an increase of 10% in printing capacity. Furthermore, deciding filling patterns is crucial in additive manufacturing as well. In general, a filling pattern with a higher density usually delivers higher physical strength but requires more material and extended model build time. Studies were conducted to utilize bio-inspired filling patterns for generating tool-paths of 3D models, where bone-like porous and honeycomb structures were investigated in [9] and [10], respectively.

Throughout the years, heuristic and meta–heuristic algorithms have been developed and widely adopted to solve combinatorial optimization problems in industrial applications [11]–[16]. Yahyaoui *et al.* [13] proposed a heuristic for solving a job–shop scheduling problem, which can yield a lower number of iterations and thus a shorter processing time. Watanabe *et al.* [14] considered the stack palletizer scheduling problem, which is NP–hard. In their work, a genetic algorithm (GA)–based solver was proposed. In [15], Lee *et al.* proposed an ACO–based control algorithm to optimize sensing schedules of individual sensors in wireless sensor networks. A hybrid algorithm combining ACO and particle swarm optimization (PSO) was proposed for robot motion control in [16].

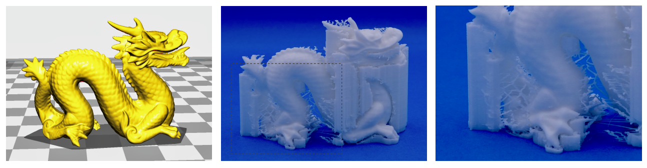


Fig. 1: (a) CAD of the model “dragon_65_tilted_large” [3]. (b) The model printed together with the supporting structure. (c) Strings found on the printed model.

Thompson and Yoon [17] developed a path planning algorithm for minimizing the amount of material wasted during a aerosol printing process. Two motion control methods were utilized in their work, including linear segments with parabolic blends and minimum time trajectory. Wah *et al.* [18] considered the optimization problem in layered manufacturing and tried different approaches to reduce time spent on transitions. In [19], the author demonstrated the possibility to tackle an rural postman problem (URPP) with ACO by first transforming it into a traveling salesman problem (TSP). In [20], Tewolde and Sheng evaluated the performance of ACO and GA in robot tool–path integration problems and showed that optimizers with ACO can yield solutions with higher quality. Researches [20]–[22] have been conducted to analyze the performance of different algorithms which are frequently used in solving TSP. The results concurred that, in general, solvers based on ACO can deliver desirable results. The selection of ACO over other meta-heuristics is sometimes more related to the problem formulation and the operations of the algorithm itself. Based on that, this work focuses on developing an ACO–based solver.

In the current work, a tailor–made ACO–based solver and a refinement process are proposed to solve the URPP in 3D printing tool–path optimization processes. The proposed solver is implemented with parallel processing capabilities to further improve its performance.

Frederickson's algorithm is a well–known approach for solving URPP and has an approximation factor of 1.5 [23]. It is similar to the approximation algorithm proposed by Christofides [24] that aims for TSP. Helsgaun's Lin-Kernighan heuristic is one of the state-of-the-art methods for finding optimal or sub-optimal solutions of TSP [25]. The method can successfully find the optimal solutions for many TSP instances up to medium scale. However, it is not implemented in this work due to its relatively high computational burden, which might prolong the whole object fabrication process rather than contributing any saving. Groves and Vuuren [26] proposed heuristic–based algorithms for general URPP. Their simulation results show that their algorithms can provide decent solutions to RPP with relatively low computational complexities. Recently, Fok *et al.* formulated the nozzle motion planning problem as a TSP and proposed a relaxation scheme for shortening the processing time without causing significant impact to the model print time [27]. However, different from ACO, Christofides' and Frederickson's algorithms cannot be benefited from parallel processing. Nonetheless, for comparison purposes, both algorithms were implemented and evaluated in the later sections of this paper.

## III. Problem Formulation

In this section, the formulation of the 3D printing tool–path optimization problem is presented. To print a 3D object, a massive number of print segments is utilized to assemble its structure. The printing nozzle traverses all print segments and deposits molten plastic filament onto them.

The tool–path integration problem is defined as to find a path for the nozzle to traverse all print segments in a 3D model. The positions and orientations of the print segments are predefined as well as the initial position of the printing nozzle. While finding a fast printing path for the nozzle, a constraint is imposed in the optimization process to discourage the nozzle from hopping among shells (*i.e.* boundaries of disjoint regions) on the same layer of the model.

The optimization problem is a tool path optimization with additional constraints. It can indeed be considered as an alternated version of URPP with constraints. Let $G = (V, E)$ be a connected and undirected graph where $V$ is a vertex set and $E$ is an edge set. The objective of an URPP solver is to find a fast tour traversing a set of required edges $E_r$, where $E_r \subset E$. An edge is formed by two extremities, *i.e.* $(i, j)$, where $i \in V, j \in V$, and $i \neq j$. A cost matrix is associated with all edges in $E$. The time cost of traversing the edge $(i, j)$ is expressed as $c_{(i,j)}$, where $c_{(i,j)} = c_{(j,i)}$ since it is symmetric, which means the costs between any two extremities are the same from either direction.

The formulation of the URPP and other notations [28] are elaborated as follow. A URPP can be solved by finding a set of edges $E_p$ of minimum total cost, such that $E_r \cup E_p$ is Euler and $E_p \subset \{(i, j) : i \in V,\ j \in V,\ i \neq j\}$. Therefore, a tour that traverses all edges in $E_r$ can be generated using the set $E_r \cup E_p$. Furthermore, let $E_y(i)$ be the set of edges that intersect at the extremity $i$. Consider $E_y(i)$ and a set of vertices $S \subset V$. Let $\Omega_y(S)$ be the cutset of $S$ with respect to $E_y(i)$. Therefore, $\Omega_y(S)$ are the edges in $E_y(i)$ such that one extremity of each edge is in $S$ and the other is in $V \setminus S$. Let $x_e$ be the count of the edge $e$ in $E_p$. The URPP can then be formulated as follows.

Minimize
$$\sum_{e \in E_D} c_e x_e, \tag{1}$$
subject to:
$$\sum_{e \in \Omega_D(v)} x_e \ \equiv\ |E_r(v)| \bmod 2,\ v \in V, \tag{2}$$
$$\sum_{e \in \Omega_D(S)} x_e \ \geq\ 2,\ S \subset V,\ \Omega_r(S) = \phi, \tag{3}$$
$$x_e \in \mathbb{Z} : x_e \geq 0,\ e \in E_d. \tag{4}$$

Here, constraint (2) establishes the Euler property of $E_r \cup E_p$ while connectivity is guaranteed by constraint (3).

In this work, tool-path optimization is transformed into a URPP with additional constraints. The tool–path optimization problem is defined on an undirected and connected graph $G = (V, E)$, where $V$ is the vertex set and $E$ is an undirected edge set which contains a subset $E_\mathrm{r}$ of the required edges, where $|E_\mathrm{r}| = n$. Here, a segment $(v_i, v_j)$ is defined as a directed path which begins with vertex $v_i$ and ends with vertex $v_j$, and vice versa. The required edges are the print segments. An edge is an undirected connection that connects two vertices. The vertex set $V$ contains all the vertices associated with edges in $E_\mathrm{r}$ and an extra vertex $v_\mathrm{st}$ which is representing the starting location of the nozzle. In order to adopt a URPP solution for a nozzle path planning problem, a virtual segment is created by using the initial position of the printing nozzle as its two extremities. This virtual segment is therefore expressed as $(v_\mathrm{st}, v_\mathrm{st})$ and it is considered as a required edge. After a solution is obtained by using URPP solvers, the tour can be transformed into a direct route for the printing nozzle by breaking the virtual segment $(v_\mathrm{st}, v_\mathrm{st})$, such that the resultant path starts with the predefined starting location of the printing nozzle and all the print segments are visited. Note that different vertices can be collocated as multiple segments can be connected to the same spot. Therefore
$$|V| = 2n + 1.$$
For typical 3D printers, the time required to traverse a print segment is independent of its connecting transitions. In all feasible print plans, filament must be deposited onto all print segments once, the total time required to traverse all print segments can be considered as a constant. Because of that, the duration for traversing print segments can be neglected in the optimization process.

A time–cost function $t$ is associated with all edges in $E$. Here, $t(v_i, v_j)$ is a time–cost function to calculate the time cost for traversing a transition $(v_i, v_j)$. For typical 3D printers, $t(v_i, v_j)$ is associated with two components, such that
$$t(v_i, v_j) = t_\mathrm{d}(v_i, v_j) + t_\mathrm{r}(v_i, v_j).$$
The first component $t_\mathrm{d}(v_i, v_j)$ denotes the time required for performing a transition, which can be calculated with a given motion model of the nozzle. The second component $t_\mathrm{r}(v_i, v_j)$ represents the time required by the extruder to perform a retraction process. Note that the cost function is symmetric, such that $t(v_i, v_j) = t(v_j, v_i)$, $\forall v_i, v_j \in V$.

In this work, the motion model proposed in [27] is adopted. While traversing a segment, the velocity of a nozzle is allowed to be changed following the given acceleration and deceleration values. It is assumed that the nozzle can stop precisely at the instructed coordinates and the corresponding time cost can be calculated with triangular and trapezoidal velocity profiles, which will be elaborated shortly. Let $a_1$ and $a_2$ be the maximum acceleration and deceleration values of the printing nozzle, respectively, and let the maximum velocity for the nozzle to traverse a transition be $v_\mathrm{max}$. In order to minimize the time cost, the nozzle tries to accelerate as much as possible when going through a transition. The minimum distance required for the nozzle to reach its maximum velocity and then to stop precisely at the instructed coordinates is calculated as
$$d_\mathrm{min} = \frac{v_\mathrm{max}^2}{2}\left(\frac{1}{a_1} - \frac{1}{a_2}\right).$$

Let $d$ be the length of a transition segment. The time cost required by the nozzle to traverse a transition can be denoted as

$$t_d = \begin{cases} \sqrt{2d\left(\frac{1}{a_1} - \frac{1}{a_2}\right)} & \text{if } d \leq d_{\min}, \\ \frac{v_{\max}}{a_1} - \frac{v_{\max}}{a_2} + \frac{d - d_{\min}}{v_{\max}} & \text{otherwise.} \end{cases}$$

Furthermore, the time spent on preforming a retraction operation is denoted as $t_r$. In general, retraction involves withdrawing filament and adjusting the level of the print bed. In this work, $t_r$ is assumed to be constant.

A feasible solution to a tool–path optimization problem is a directed path that leads the printing nozzle through all required edges at least once starting from a predefined starting vertex. Meanwhile, the ending vertex is not necessary to be equivalent to the starting vertex. The nozzle does not return to its starting location. The nozzle stops or moves to next layer right after it traverses all required edges on the current layer.

## IV. Proposed Tool–Path Optimizer

In this paper, a 2–layers optimizer is proposed to solve the tool–path optimization problem in additive manufacturing. The proposed optimizer further utilized a solver based on ACO, a nature inspired meta–heuristic to handle TSP and URPP in the optimization process. The proposed optimizer aims to improve the visual quality by alleviating the strings phenomenon on printed models. Meanwhile, the optimizer also improves the efficiency of the printing process by minimizing the time spent on traversing transitions.

The proposed optimizer begins with the input of print segments that generated by a slicer software. As mention in Section II, in this paper, the process for generating print segments from CAD model is handled by the slicer software and it is considered as out of scope. The proposed optimizer then searches for a desirable route to traverse all given print segments in the model. Note that during the search, to ensure the integrity of the printed model, none of the original print segments is reoriented or skipped. Therefore, apart from alleviating the strings issue, accuracy and the physical properties of the printed model should not be affected by the proposed optimizer in general.

The proposed optimizer has a 2–layer architecture, which is illustrated in Fig. 2. It takes a sliced model as its input. Each sliced layer is then further dissected into disjoint parts based on their shells which will then be optimized individually. Details on the designs and the operations of the proposed optimizer are elaborated as follows.

### A. Parts Visiting Sequence

Every time when the print nozzle hops between disjoint parts, strings could be generated. Therefore, the proposed optimizer aims to suppress the printing nozzle from conducting unnecessary hoppings among shells. It searches for a fast path to traverse all isolated parts on the layer one by one which does not cross the boundary of a disjoint part until all print segments in that part are deposited.

To find such a path, the proposed optimizer transforms the problem into a TSP by conducting the following steps. Each disjoint part is considered as a single node. The time–cost between two nodes is estimated using the shortest distance from any two print segments from the two parts involved. An extra node is then added which represents the starting location of the nozzle for the current layer. Based on that, the problem becomes finding a fast path that visits all nodes exactly once. The result can be obtained using a TSP–solver, which will be introduced in the next section.

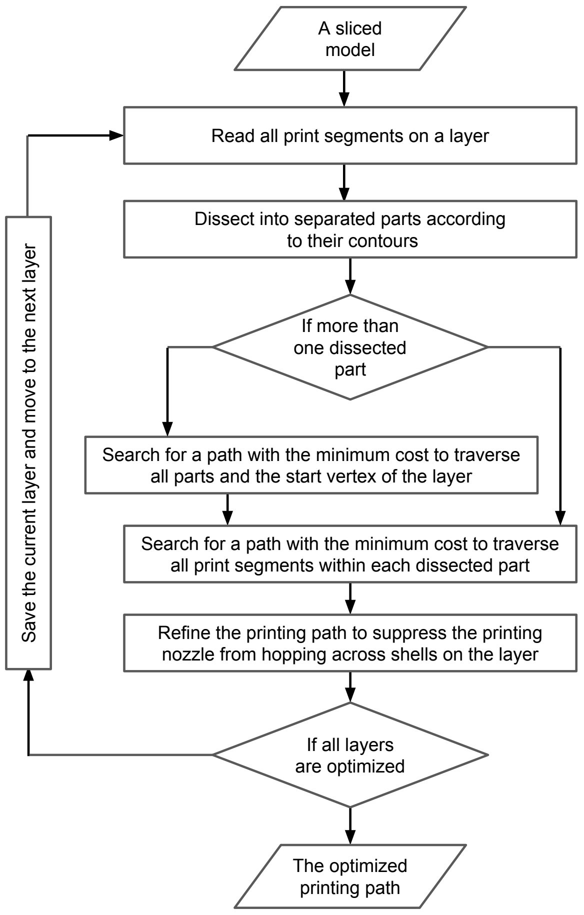

A sliced model
Read all print segments on a layer
Dissect into separated parts according to their contours
If more than one dissected part
Search for a path with the minimum cost to traverse all parts and the start vertex of the layer
Search for a path with the minimum cost to traverse all print segments within each dissected part
Refine the printing path to suppress the printing nozzle from hopping across shells on the layer
If all layers are optimized
The optimized printing path
Save the current layer and move to the next layer

Fig. 2: The workflow of the proposed optimizer.

After finding a fast parts visiting sequence, all disjoint parts are connected with a route. An illustrative example is shown in Fig. 3, which has 18 parts connected with transitions rendered in red lines. The two vertices in each part that are connecting to their adjacent parts on the visiting sequence are denoted as the local start and end points of the part under consideration. These two vertices are used in the next stage when optimizing the printing path inside each part.

### B. Print Segments Visiting Sequence

The proposed optimizer considers the printing path for each isolated part separately. The optimizer begins with forming a virtual segment between the local start and end points that obtained from the previous stage. Hence, the problem becomes finding a fast path traversing all required segments, including the print segments and the virtual segment inside the part, which is an URPP. The optimizer utilizes an URPP–solver to search for a tour that accomplishes the above condition. The

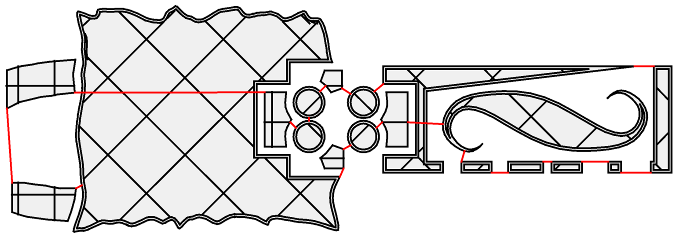


Fig. 3: An illustration of a layer of the model "3DHackerTest" [3]. The black lines represents the print segments. Interiors of each part is shaded in grey color. The red lines represent the transitions that connect disjoint parts.

proposed ACO–based URPP solver and other solvers under test will be introduced and explained in the next section. After obtaining a tour that traverse all the required segments, a directed path, that begins and finishes at the given local start and end points correspondingly, is constructed by breaking the virtual segment.

## *C. Refinement Process*

A refinement process is then executed on the directed path of each disjoint part obtained above to stop the nozzle from hopping across its shell, which is common for concave parts or parts with hollow structures. The refinement process in the proposed optimizer detects and replaces these unnecessary shell–crossing transitions with the fastest consecutive transitions that traverse inside the part. Conventional path finding algorithms [29] can be used to achieve such a goal. With the results obtained from the process mentioned in Sections IV-A, IV-B, and IV-C, printing paths of disjoint parts on the current layer can be integrated into a single route.

## V. Problem Solvers

## *A. Christofides' and Frederickson's Algorithms*

Christofides proposed a polynomial time algorithm for solving TSP in [24]. This deterministic algorithm is proven to have an approximation factor of 1.5 in solving TSP [24]. Later, Frederickson's algorithm [23] was proposed for solving URPP. Many similarities can be observed between Christofides' algorithm and Frederickson's algorithm. They both share three major procedures in finding a solution, namely, constructing a minimum spanning three (MST), performing a minimum perfect matching, and shortcuts searching. These two algorithms are utilized as references in the performance evaluation section of this work. Due to their similarities, only Frederickson's algorithm is explained below. Readers who are interested in Christofides' algorithm may refer to [24].

Frederickson's algorithm begins with a given undirected and connected graph $G = (V, E)$ and a set of required edges $E_\text{r} \in E$. A MST is then constructed to connect all the edges in $E_\text{r}$. Those new edges introduced in the MST construction process form a new set $E_\text{mst}$. A minimum perfect matching is then conducted on the sumset $E_\text{r} + E_\text{mst}$ to connect vertices with odd degrees. Those extra edges added in the matching process then form another new set $E_\text{matching}$. An Eulerian tour can then be found in the sumset $E_\text{r} + E_\text{mst} + E_\text{matching}$. Consecutive edge pairs on the tour, whose edges are not in $E_r$, are replaced with shortcuts to further optimize the tour.

## *B. Ant Colony Optimization*

ACO is a well–known meta–heuristic which is inspired by the behaviour of ants. It was first proposed to solve TSP [30]. When solving TSP, each artificial ant searches for a path composed of consecutive edges, which will then be evaluated. Pheromone is then deposited on paths discovered by the ants. The amount of pheromone deposited on a path is proportional to its evaluation result or inversely proportional to its cost. As a result, a pheromone table is constructed. The pheromone level associated with an edge $(i, j)$ is indicated as $\tau_{i,j}$, given that $i$ and $j$ are the two vertices of that edge. In the next iteration, the whole process repeats. However, ants will then make their search decisions based on both heuristic and pheromone information. In general, ants tend to choose an edge with a higher pheromone level. The probability of choosing a path $(i, j)$ is expressed as

$$p^k_{i,j}(t) = \frac{[\tau_{i,j}(t)]^\alpha [\eta_{i,j}]^\beta}{\sum_{l \in N^k_i} [\tau_{i,l}(t)]^\alpha [\eta_{i,l}]^\beta}. \tag{5}$$

Given that the $k$-th ant is currently at vertex $i$ while constructing its path. The heuristic value between two vertices $v_i$ and $v_j$ is denoted as $\eta_{i,j}$. Here, $\eta_{i,j}$ is inversely proportional to the distance between $v_i$ and $v_j$. Furthermore, $N^k_i$ denotes a vertex set that includes all vertices which can form valid paths with vertex $i$ and have not been visited by the $k$-th ant yet. Variables $\alpha$ and $\beta$ are control parameters used in generic ACO that can affect the quality of solutions generated. Further information on $\alpha$ and $\beta$ can be found in [30]. At the end of each iteration, the levels of pheromone on all edges are reduced in an operation called evaporation. The pheromone level $\tau_{i,j}$ at the end of the $t$ iteration is updated as

$$\tau_{i,j}(t+1) = (1-\rho)\tau_{i,j}(t) + \sum_{k=1}^{m} \Delta\tau^k_{i,j}(t).$$

Here, parameter $\rho$ controls the rate of pheromone evaporation, where $\rho \in (0, 1)$. Here, $m$ is the number of ants. The amount of pheromone deposits by the $k$-th ant on edge $(i, j)$ is $\Delta\tau^k_{i,j}(t)$. The value of $\Delta\tau^k_{i,j}(t)$ is proportional to the quality of the path, or inversely proportional to the cost of the path. As ACO iterates, pheromone concentrations on certain edges, which are associated with paths with good quality, will remain high. In this work, (1) is utilized as the fitness function for ants to evaluate the quality of a path.

## *C. Modified Ant Colony Optimization*

In the proposed modified ACO, a stochastic-based segment integration process is implemented and integrated into the generic ACO solver. In such process, segment-transition-segment (STS) groups that are more preferred by ants will be integrated to avoid further changes.

During a search process in ACO, frequently visited STS groups deposited with high concentration of pheromone are very likely to be included in good solutions. From our studies, STS groups found in print plans are usually associated with short transitions. The rationales behind such observation are that as print segments are normally expressed as straight lines in typical 3D printing applications, curvy contours are constructed using multiple segments chained up with short transitions. Therefore, it is common for print plans to comprise large volumes of tiny transitions and lead to an unnecessary big search space.

The segment integration process begins at the end of each iteration of the ACO process. With the best solution obtained so far, STS groups associated with such solution are analyzed and selected to be integrated through a stochastic mechanism. Results are then used to update and shrink the search space. Ants in the following iterations will then proceed their search on the trimmed search space.

Details of the process are elaborated as follows. At the end of an iteration, the best solution discovered so far, namely $S_a$, is expressed as

$$S_a = \langle v_{\text{st}}, v_{a_1}, \cdots, v_{a_{2i-1}}, v_{a_{2i}}, v_{a_{2(i+1)}}, v_{a_{2(i+1)+1}}, \cdots, v_{a_{2n}} \rangle.$$

Here, $S_a$ contains $n$ transitions to connect all $n$ print segments to $v_{\text{st}}$. The $i$-th and $(i+1)$-th print segments are $(v_{a_{2i-1}}, v_{a_{2i}})$ and $(v_{a_{2(i+1)}}, v_{a_{2(i+1)+1}})$, respectively. Furthermore, $(v_{a_{2i}}, v_{a_{2i+}})$ represents the $i$-th transition, which connects the $i$-th and $(i+1)$-th print segments.

In iteration $t$, the pheromone and heuristic values of the $i$-th transition in the STS group joining the $i$-th and $(i+1)$-th segments are $\tau_{2i,2i+1}(t)$ and $\eta_{2i,2i+1}$, respectively. These two values together with the control parameters $\alpha$ and $\beta$ form an evaluation function of the $i$-th transition and it is expressed as $[\tau_{2i,2i+1}(t)]^{\alpha}[\eta_{2i,2i+1}]^{\beta}$. STS groups with high evaluation values will have higher probabilities to be integrated in the process.

The probability for a STS group, which contains the $i$-th transition, to be integrated is expressed as

$$p_{(i,i+1)}(t) = \min\left(\frac{\theta n[\tau_{2i,2i+1}(t)]^{\alpha}[\eta_{2i,2i+1}]^{\beta}}{\sum_{j=1}^{n}[\tau_{2j,2j+1}(t)]^{\alpha}[\eta_{2j,2j+1}]^{\beta}}, 1\right).$$

The physical meaning of $\theta$ is the expected ratio of STS groups to be integrated at the end of the current iteration, where $\theta \in [0,1]$. Note that the expected number of STS groups to be integrated at the end of the $t$–iteration is calculated as

$$\sum_{i=1}^{n} p_{(i,i+1)}(t) = \sum_{i=1}^{n}\left[\min\left(\frac{\theta n[\tau_{2i,2i+1}(t)]^{\alpha}[\eta_{2i,2i+1}]^{\beta}}{\sum_{j=1}^{n}[\tau_{2j,2j+1}(t)]^{\alpha}[\eta_{2j,2j+1}]^{\beta}}, 1\right)\right].$$

Since $\theta \in [0,1]$, we have

$$\sum_{i=1}^{n} p_{(i,i+1)}(t) \leq \sum_{i=1}^{n}\left(\frac{\theta n[\tau_{2i,2i+1}(t)]^{\alpha}[\eta_{2i,2i+1}]^{\beta}}{\sum_{j=1}^{n}[\tau_{2j,2j+1}(t)]^{\alpha}[\eta_{2j,2j+1}]^{\beta}}\right) = \theta n.$$

Therefore, the expected number of STS groups to be integrated is upper bounded by $\theta n$. With the increase of $\theta$, it is expected that more STS groups will be integrated. Print plans usually comprise segments located close to each other. Due to such property, their corresponding STS groups are often associated with high pheromone and heuristic values, which are very likely to be selected by ants as part of the final solution. The proposed method tries to integrate those segments and shrinks the problem scale as the optimization process iterates. With the reduction in the problem size, ants can perform a more thorough search in the shrunk search space which helps to find better solutions when comparing to generic ACO using the same amount of ants. Besides, the value of $\theta$ can also affect the computational time of the proposed optimizer. A high value of $\theta$ leads to more STS group integrations and thus shrinks the search space more rapidly. If the value of $\theta = 0$, the behavior of the proposed ACO solver is identical to the generic ACO solver. On the contrary, if $\theta \to 1$, it can be expected that a high proportion of STS groups will be integrated. It is possible that some good STS groups could be eliminated unintentionally during the process which will result in a degradation of solution quality. According to our empirical data, the proposed method can yield desirable results when $\theta \in [0.2, 0.4]$.

## VI. Experiments

### A. Experiment Settings

Experiments were conducted to evaluate the performance of the proposed optimizer and the modified ACO. In the experiments, a slicer software Cura–15.04.6 [2] was utilized to slice 3D models with its default settings. Furthermore, retraction was enabled, vertical hop and retraction length are 0.075 mm and 4.5 mm, respectively. The slicer used 10% filling density in the infilling process. In the experiments, 8 CAD models [3], [31] with different unique characteristics were selected for testing. It is worth to note that, for some of the models, the number of segments on a layer are larger than seven thousand. Different path optimization modules were utilized for comparison purposes. Cura is integrated with a greedy–based optimizer [2], which is used as the reference in the evaluations. In the experiments, three optimizers were utilized separately to further optimize print plans obtained from Cura, including

1) Christofides' and Frederickson's algorithms (also referred to CF here onwards),
2) a generic ACO, and
3) the proposed modified ACO.

For 1), Christofides' algorithm is employed in solving the TSP in arranging parts visiting sequence and Frederickson's algorithm is utilized in solving the URPP in arranging print segments visiting sequence. For 2) and 3), the same optimizer was employed in solving both TSP and URPP in the whole optimization process. In the experiments, the estimated print times of print plans were analyzed using GCodeAnalysor–1.1 [32]. All programs were executed on a computer with an Intel Core i7 3.6 GHz processor, 16 GB RAM, and Windows 10 operating system. Search processes in ACO can be speeded up with the help of parallel processing [16]. All ACO-based solvers in this work were implemented in a multi-threads manner to harness the full processing power of the CPU. In the experiments, due to the stochastic nature of ACO, ACO–based optimizers tend to yield better solutions with an increase of ants and iterations. To give a fair comparison with

TABLE I: PARAMETERS UTILIZED IN GENERIC ACO, THE PROPOSED MODIFIED ACO-BASED OPTIMIZERS, AND CURA.

| | |
|---|---|
| Number of iterations | 8 |
| Number of ants | 8 |
| $\alpha$ | 1 |
| $\beta$ | 5 |
| $\rho$ | 0.5 |
| $\theta$ | 0.2 |
| Layer height | 0.1 mm |
| Z-hop retracting | 0.075 mm |
| Retraction speed | 40 mm/s |
| Retraction length of filament | 4.5 mm |
| Travel speed | 150 mm/s |
| Print speed | 50 mm/s |
| Filling density | 10% |
| Print temperture | 220 °C |
| Bed temperture | 70 °C |

Christofides' and Frederickson's algorithms, the number of ants and iterations for ACO–based optimizers are both set to 8 such that the processing time required by ACO–based optimizers are close to that of Christofides' and Frederickson's algorithms. All ACO-related parameters used in this work were chosen based on [20], which demonstrated promising results in tackling a similar path search problem. Also, our preliminary study showed that the ACO optimizer that utilized the parameter sets in [20] outperforms those utilized other reasonable parameter sets.

A summary of the parameters used in this work is shown in Table I.

In the evaluations, two parameters were chosen as performance indicators, namely the post processing time and the estimated print time. Furthermore, to verify the accuracy of the estimated print time, 4 out of the 8 models were printed using a 3D printer and their corresponding print times were recorded. The visual qualities of the 4 printed models were further examined based on the amount of strings formed on their surfaces.

## B. Experiment Results

*1) Accuracies:* Experiments were conducted to evaluate the dimensional accuracy of printed models using print plans optimized by the modified ACO.

Similar to [33], in this set of experiments, the model "testcube_25mm" [34] with 20% filling density was utilized, which is a cube with edge length equals 25mm. The model was printed using print plans generated directly from Cura and also other optimized plans generated using the aforementioned algorithms. The vertices of the cube were labelled as shown in Fig. 4.

In this work, dimensional accuracy was evaluated by the average absolute differences between edge lengths of the printed cubes and the corresponding CAD model. The measurements are shown in Fig. 4, which were obtained using a "Mitutoyo Digital Caliper 500-196-20".

According to Fig. 4, the average absolute differences (%) of edge lengths for models generated by Cura, CF, ACO, and the modified method are 0.1225%, 0.1150%, 0.1333%, and 0.1142%, respectively. Results suggested that the proposed method can yield printed models with similar dimensional accuracies as those obtained from other methods under test.

*2) Post–processing time:* All optimizers under test were executed 50 times to obtain their average run times. Results are given in TABLE II. According to the results, when the parameters of the modified ACO optimizer was configured as shown in TABLE I, it yields comparable post–processing times as Christofides' and Frederickson's algorithms. Furthermore, when comparing with the generic ACO optimizer under the same configurations, the proposed one can reduce the overall post–processing time significantly by 34.93%.

*3) Estimated Print Time:* Results obtained using the GCodeAnalysor are presented in TABLE III. It can be observed that print plans from all optimizers under test can yield shorter estimated print time than those directly from Cura. Moreover, the proposed optimizer can generate print plans with the shortest estimated print time among the others. It obtained a maximum saving of 8.58% in estimated print time on the model "hold_test" versus Cura.

As mentioned in Section V-A, Christofides' and Frederickson's algorithms are deterministic algorithms. Therefore, the optimizer with CF always generates identical results. According to TABLE III, when comparing the maximum and the mean values of results obtained using modified ACO and CF, the modified ACO can yield lower values in all tested models.

Performances of the modified ACO and the generic ACO on each 3D model are compared separately. The mean and standard deviations (SD) of the estimated print times of model "dragon_65_tilted_large" using the modified ACO are denoted by $\mu'_{\text{ACO}}$ and $\sigma'_{\text{ACO}}$, respectively. Similarly, the mean and SD of the estimated print time using the generic ACO are represented respectively by $\mu_{\text{ACO}}$ and $\sigma_{\text{ACO}}$. The difference between the two means $(\mu'_{\text{ACO}} - \mu_{\text{ACO}}) = -41$ with the corresponding SD $= 0.6681$. The 99% confidence interval for $(\mu'_{\text{ACO}} - \mu_{\text{ACO}})$ is therefore $(-42.76, -39.24)$, which suggests that the modified ACO is likely to yield shorter estimated print times for model "dragon_65_tilted_large". Similar analyses were conducted on all other models. Results suggest that the modified ACO tends to yield shorter estimated print times than the generic ACO in 7 out of 8 models. In addition, by considering the savings in post-processing time mentioned in the last experiment, the results further suggest that the modified ACO can deliver better performance on accelerating the printing process and requires significantly shorter post-processing time than the generic ACO.

*4) Actual print time:* Among the 8 models under test, 4 of them were further selected to go through an actual printing experiment. The printing experiment was conducted on a WiseMaker 3D Printer W250 [1]. Print plans obtained from different optimizers were printed, while the whole process was recorded and timed. Their corresponding actual print time were presented in TABLE IV. In our measurements, pre–heating times for both the nozzle and the print bed were excluded. The timer was started once the nozzle moved away from its default location and was stopped when the nozzle finished the last print segment and returned to its default location. All results reported in TABLE IV concur with those observations

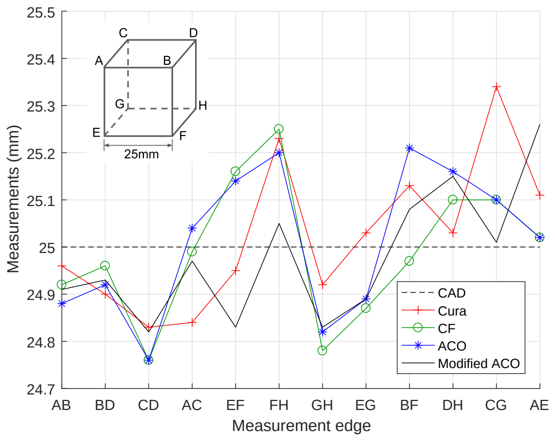


Fig. 4: An illustration (top left) of the labelled cube [34] used in the experiment and the edge measurements of the printed models using print plans from different optimizers. The dotted line represents the edges' length of the CAD model.

in the last section, which verify the accuracy of the estimated print time presented. The proposed optimizer again yields the best results when comparing with its counterparts.

*5) Visual quality:* The visual quality were verified according to the amount of strings, which are residual materials left on the surface of a printed model. Pictures of the model "hold_test" [31] fabricated using print plans from different optimizers are shown in Figs. 5. In this experiment, only residual materials with any of its dimensions longer than 0.5mm are regarded as strings and they are highlighted with red boxes.

According to Fig. 5, the number of strings identified on models printed using print plans from Cura, CF, ACO, and the proposed method are 23, 4, 6, and 2, respectively. Even with the retraction function enabled, the model from the print plan generated by Cura has a considerable amount of strings on surface. In contrast, by allowing the nozzle to hop across boundaries only when necessary, the proposed optimizer can effectively alleviate the strings issue in its printed models. Print plans optimized together with the proposed refinement process, regardless of the solvers being used, can always reduce the number of strings in their printed outcomes and yield shorter actual print time than those obtained from Cura.

## VII. Discussion

According to the post–processing times given in TABLE II, it can be observed that when optimizing print plans with different complexities (*i.e.* number of print segments), the time required by the generic ACO solver increases faster comparing to that of the proposed solver. The proposed solver speeds up its computation by shrinking the search space in each iteration via its segment integration process. As mentioned in Section V-C, an immature convergence may lead to incorrect STS grouping and degrade the solution quality. Based on the estimated print time given in TABLE III, such modifications to the ACO solver did not degrade solution quality when comparing to that of the generic ACO. During a segment integration process, while a STS group that is more preferred by ants is being integrated, this action can eliminate many other non-preferred alternatives. With desirable STS groups being preserved and having the search space being shrunk over time, the proposed ACO solver is more efficient in obtaining high quality solutions.

When comparing the performance of optimizers using the proposed ACO solvers to those using Christofides' and Frederickson's algorithms, insignificant deviations can be observed in their estimated print time and post-processing time. Apart from the reasons mentioned above, the native parallel nature of ACO makes it possible to be executed in parallel with the help of modern multi-core CPU and greatly shorten its post–processing time. According to TABLE II, it is observed that optimizers with the proposed ACO solver scale well with the problem size (*i.e.* number of segments in the models). Nevertheless, it has the potential to further shorten its computational time when more processing units, such as graphics processing units (GPUs), are available. From another point of view, with the extra parallel processing power, optimizers with the proposed ACO solver can utilize more ants to perform parallel searches and yield better solutions within the same amount of time. On the contrary, Christofides' and Fredericksson's algorithms cannot be benefited from parallel processing as these two algorithms consist of sequential procedures as mentioned in Section V-A.

## VIII. Conclusion

In this paper, an ant colony optimization based tool–path optimizer is proposed for finding desirable tool–paths in 3D printing applications, which can both shorten actual print time and improve visual quality of printed objects. The proposed optimizer has a 2–layers structure that formulates the optimization process as travelling salesman and undirected rural postman problems, correspondingly. A modified ACO solver is proposed and utilized in the optimizer to tackle both problems. By exploiting the unique properties in 3D printing and taking the advantages of parallel processing, the proposed ACO solver can provide decent solutions within a reasonable processing time. Simulation and experiment results shown that the proposed optimizer together with the proposed ACO solver can yield shorter model build time than other optimizers under test. Furthermore, experiment results also verify the effectiveness of the refinement processes in the proposed optimizer in alleviating the strings phenomenon known in 3D printing applications. The proposed optimizer, which comprises a modified ACO-based URPP solver and a complete workflow, can speed up the printing process and improve the visual quality of the output simultaneously.

## References

[1] "WiseMaker W Series 3D printers | Movehand.com," (Accessed: 2017-02-15). [Online]. Available: http://movehand.com/w140-3d-printer

[2] "Cura 3D printing slicing software," (Accessed: 2017-02-20). [Online]. Available: https://ultimaker.com/en/products/cura-software

[3] "GitHub - Ultimaker/CuraEngine," (Accessed: 2017-02-22). [Online]. Available: https://github.com/Ultimaker/CuraEngine/tree/4c547b9a66433885d4d4128f5f416982878b7e56/tests/allround_test

[4] A. Armillotta, "Assessment of surface quality on textured FDM prototypes," *Rapid Prototyping Journal*, vol. 12, no. 1, pp. 35–41, 2006.

TABLE II: POST–PROCESSING TIMES (s).

| Models | CF | | | | ACO | | | | Modified ACO | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Mean | Max | Min | SD | Mean | Max | Min | SD | Mean | Max | Min | SD |
| 3DHackerTest | 212.17 | 214.90 | 211.62 | 0.55 | 284.96 | 348.11 | 214.18 | 27.89 | 215.98 | 293.04 | 155.71 | 34.80 |
| ctrlV_3D_test | 496.24 | 498.38 | 494.23 | 0.94 | 615.12 | 725.47 | 572.36 | 39.69 | 495.97 | 582.58 | 427.45 | 48.16 |
| Debailey_x10 | 170.06 | 170.26 | 169.89 | 0.07 | 287.12 | 297.99 | 280.31 | 4.36 | 180.85 | 190.24 | 175.61 | 3.57 |
| dragon_65_tilted_large | 312.91 | 314.11 | 312.56 | 0.30 | 457.39 | 466.07 | 450.42 | 3.51 | 279.69 | 287.62 | 274.77 | 3.24 |
| testModel | 17.77 | 17.91 | 17.67 | 0.05 | 76.72 | 78.38 | 74.88 | 0.84 | 51.41 | 53.75 | 48.57 | 1.22 |
| TortureTestV2 | 49.08 | 49.30 | 48.75 | 0.11 | 120.18 | 121.34 | 118.95 | 0.59 | 82.62 | 83.26 | 82.19 | 0.27 |
| UltimakerRobot_ support_2015 | 157.95 | 158.63 | 157.77 | 0.15 | 261.21 | 263.14 | 259.33 | 0.97 | 153.25 | 156.34 | 151.40 | 0.89 |
| holdtest | 983.54 | 1135.23 | 954.42 | 26.58 | 1111.28 | 1287.40 | 1064.40 | 39.40 | 507.52 | 555.35 | 489.54 | 14.15 |

TABLE III: ESTIMATED PRINT TIMES (s).

| Models | Cura | CF | ACO | | | | Modified ACO | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mean | Max | Min | SD | Mean | Max | Min | SD |
| 3DHackerTest | 6551 | 6038 | 6032 | 6036 | 6028 | 1.96 | 6025 | 6030 | 6017 | 2.26 |
| ctrlV_3D_test | 14791 | 14210 | 14170 | 14176 | 14156 | 4.15 | 14169 | 14181 | 14153 | 6.51 |
| Debailey_x10 | 20672 | 19498 | 19465 | 19471 | 19460 | 2.70 | 19434 | 19441 | 19426 | 2.98 |
| dragon_65_tilted_large | 18833 | 17693 | 17669 | 17676 | 17663 | 2.95 | 17628 | 17637 | 17622 | 3.69 |
| testModel | 22578 | 22197 | 22133 | 22144 | 22122 | 4.69 | 22094 | 22102 | 22086 | 4.35 |
| TortureTestV2 | 23663 | 23332 | 23303 | 23316 | 23294 | 5.24 | 23250 | 23268 | 23238 | 6.41 |
| UltimakerRobot_support_2015 | 19872 | 19173 | 19126 | 19131 | 19120 | 2.60 | 19091 | 19098 | 19084 | 3.37 |
| hold_test | 3578 | 3363 | 3273 | 3281 | 3266 | 3.73 | 3271 | 3278 | 3263 | 3.42 |

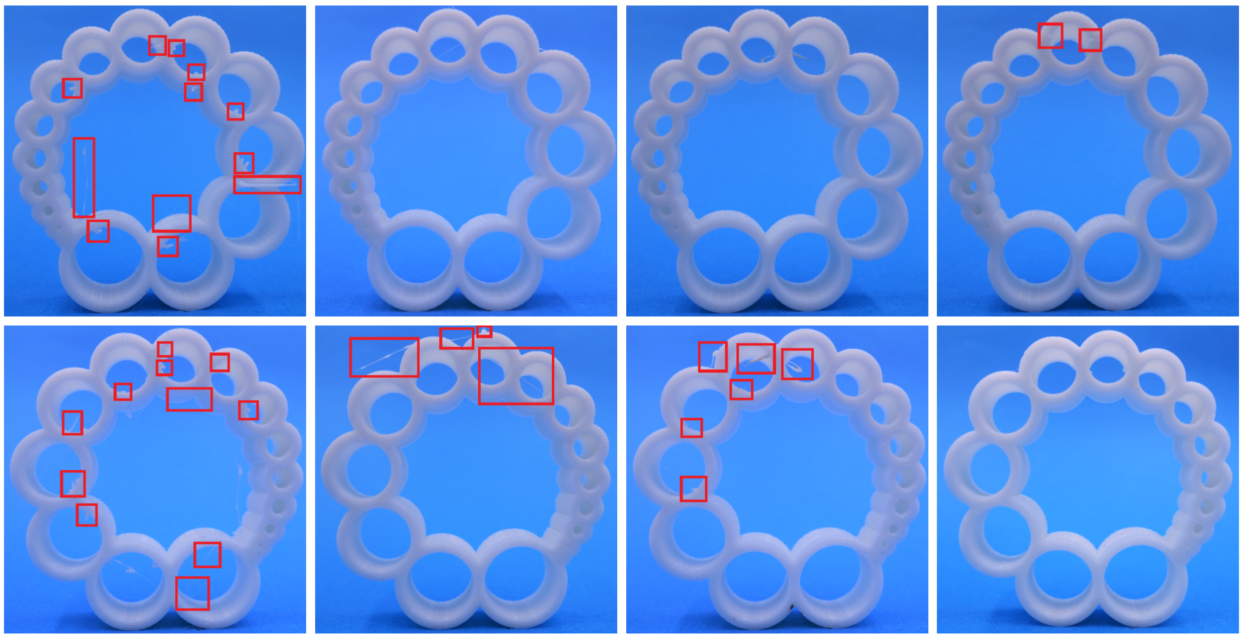

(a) Cura (b) CF (c) ACO (d) Modified ACO

Fig. 5: Top side (upper row) and bottom side (lower row) of the printed models "hold_test" optimized using different algorithms. Regions populated with *strings* (>0.5 mm) were highlighted in red boxes. Each string is highlighted and counted on either the upper or lower row to avoid double counting.

TABLE IV: ACTUAL PRINT TIMES (s).

| Models | Cura | CF | ACO | Modified ACO |
|---|---|---|---|---|
| ctrlV_3D_test | 15311 | 14657 | 14469 | 14463 |
| dragon_65_tilted_large | 18742 | 17694 | 17645 | 17013 |
| UltimakerRobot_ support_2015 | 19763 | 19209 | 19461 | 19088 |
| hold_test | 3872 | 3376 | 3406 | 3301 |

[5] M. K. Agarwala, V. R. Jamalabad, N. A. Langrana, A. Safari, P. J. Whalen, and S. C. Danforth, "Structural quality of parts processed by fused deposition," *Rapid Prototyping Journal*, vol. 2, no. 4, pp. 4–19, 1996.

[6] S. Lim, R. Buswell, T. Le, S. Austin, A. Gibb, and T. Thorpe, "Developments in construction-scale additive manufacturing processes," *Automation in Construction*, vol. 21, pp. 262 – 268, 2012.

[7] Y.-a. Jin, Y. He, G.-h. Xue, and J.-z. Fu, "A parallel-based path generation method for fused deposition modeling," *The International Journal of Advanced Manufacturing Technology*, vol. 77, no. 5, pp. 927–937, Mar 2015.

[8] J. P. Freens, I. J. Adan, A. Y. Pogromsky, and H. Ploegmakers, "Automating the production planning of a 3D printing factory," in *Winter Simulation Conference (WSC), 2015.* IEEE, 2015, pp. 2136–2147.

[9] J. Wu, N. Aage, R. Westermann, and O. Sigmund, "Infill optimization for additive manufacturing–approaching bone-like porous structures," *IEEE Transactions on Visualization and Computer Graphics*, 2017.

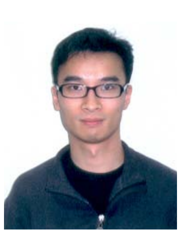
[10] A. Kvalsvig, X. Yuan, J. Potgieter, and P. Cao, "3D printing of fibre reinforced honeycomb structured composite materials," in *Mechatronics and Machine Vision in Practice (M2VIP), 2016 23rd International Conference on.* IEEE, 2016, pp. 1–6.

[11] V. Roberge, M. Tarbouchi, and G. Labonte, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," *IEEE Trans. on Industrial Informatics*, vol. 9, no. 1, pp. 132–141, Feb 2013.
[12] J. J. Kim and J. J. Lee, "Trajectory optimization with particle swarm optimization for manipulator motion planning," *IEEE Trans. on Industrial Informatics*, vol. 11, no. 3, pp. 620–631, June 2015.
[13] A. Yahyaoui, N. Fnaiech, and F. Fnaiech, "A suitable initialization procedure for speeding a neural network job-shop scheduling," *IEEE Trans. on Industrial Electronics*, vol. 58, no. 3, pp. 1052–1060, April 2011.
[14] M. Watanabe, M. Furukawa, A. Mizoe, and T. Watanabe, "GA applications to physical distribution scheduling problem," *IEEE Trans. on Industrial Electronics*, vol. 48, no. 4, pp. 724–730, 2001.
[15] J. W. Lee, B. S. Choi, and J. J. Lee, "Energy-efficient coverage of wireless sensor networks using ant colony optimization with three types of pheromones," *IEEE Trans. on Industrial Informatics*, vol. 7, no. 3, pp. 419–427, Aug 2011.
[16] H. C. Huang, "A taguchi-based heterogeneous parallel metaheuristic ACO-PSO and its FPGA realization to optimal polar-space locomotion control of four-wheeled redundant mobile robots," *IEEE Trans. on Industrial Informatics*, vol. 11, no. 4, pp. 915–922, Aug 2015.
[17] B. Thompson and H.-S. Yoon, "Efficient path planning algorithm for additive manufacturing systems," *IEEE Trans. on Components, Packaging and Manufacturing Technology*, vol. 4, no. 9, pp. 1555–1563, 2014.
[18] P. K. Wah, K. G. Murty, A. Joneja, and L. C. Chiu, "Tool path optimization in layered manufacturing," *IIE Transactions*, vol. 34, no. 4, pp. 335–347, 2002.
[19] M.-L. Pérez-Delgado, *Solving an arc-routing problem using artificial ants with a graph transformation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 241–246.
[20] G. S. Tewolde and W. Sheng, "Robot path integration in manufacturing processes: Genetic algorithm versus ant colony optimization," *IEEE Trans. on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 38, no. 2, pp. 278–287, March 2008.
[21] S. Alhamdy, A. N. Noudehi, and M. Majdara, "Solving traveling salesman problem (tsp) using ants colony (aco) algorithm and comparing with tabu search, simulated annealing and genetic algorithm," *J. Appl. Sci. Res*, vol. 8, no. 1, pp. 434–440, 2012.
[22] H. Afaq and S. Saini, "On the solutions to the travelling salesman problem using nature inspired computing techniques," *International Journal of Computer Science Issues*, vol. 8, no. 2-4, pp. 326–334, 2011.
[23] G. N. Frederickson, "Approximation algorithms for some postman problems," *Journal of the ACM*, vol. 26, no. 3, pp. 538–554, 1979.
[24] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," DTIC Document, Tech. Rep., 1976.
[25] K. Helsgaun, "An effective implementation of the lin–kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.
[26] G. Groves and J. Van Vuuren, "Efficient heuristics for the rural postman problem," *ORiON*, vol. 21, no. 1, pp. 33–51, 2005.
[27] K.-Y. Fok, C.-T. Cheng, C. K. Tse, and N. Ganganath, "A relaxation scheme for TSP-based 3D printing path optimizer," in *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, October 2016, pp. 382–385.
[28] E. Fernández, O. Meza, R. Garfinkel, and M. Ortega, "On the undirected rural postman problem: Tight bounds based on a new formulation," *Operations Research*, vol. 51, no. 2, pp. 281–291, 2003.
[29] N. Ganganath, C.-T. Cheng, and K. T. Chi, "A constraint-aware heuristic path planner for finding energy-efficient paths on uneven terrains," *IEEE Trans. on Industrial Informatics*, vol. 11, no. 3, pp. 601–611, 2015.
[30] M. Dorigo and G. Di Caro, "Ant colony optimization: a new metaheuristic," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 2. IEEE, 1999, pp. 1470–1477.
[31] R. Flórez, "Hole Test by magramo - Thingiverse," (Accessed: 2018-01-11). [Online]. Available: http://www.thingiverse.com/thing:2380801
[32] K.-Y. Fok, "GCodeAnalysor-1.1 by kyfok - Thingiverse," (Accessed: 2017-07-08). [Online]. Available: http://www.thingiverse.com/thing:1870254
[33] D. Roberson, D. Espalin, and R. Wicker, "3D printer selection: A decision-making evaluation and ranking model," *Virtual and Physical Prototyping*, vol. 8, no. 3, pp. 201–212, 2013.
[34] "Calibration Cube Collection by thingster - Thingiverse," (Accessed: 2018-01-11). [Online]. Available: http://www.thingiverse.com/thing:56671

**Kai-Yin Fok** (S'16) received the B.Sc.(Hons.) degree in Internet and multimedia technologies in 2014, and is currently working toward the Ph.D. degree in the Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Hong Kong, where he was a Research Assistant from 2014 to 2015. His research interests include sensing technologies, machine learning applications, and optimization techniques in additive manufacturing.

**Chi-Tsun Cheng** (S'07–M'09) received the B.Eng. and M.Sc. degrees from the University of Hong Kong in 2004 and 2005, respectively, and the Ph.D. degree from the Hong Kong Polytechnic University in 2009. From 2010 to 2011, he was a Post-Doctoral Fellow at the Department of Electrical and Computer Engineering, the University of Calgary. From 2012 to 2018, he was a Research Assistant Professor at the Department of Electronic and Information Engineering, the Hong Kong Polytechnic University. Since June 2018, he has been a Senior Lecturer at the Department of Manufacturing, Materials and Mechatronics, RMIT University, Melbourne, Australia. He serves as Associate Editors for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II and IEICE NONLINEAR THEORY AND ITS APPLICATIONS. His research interests include Wireless Sensor Networks, Internet of Things, Industry 4.0 Technologies, Cloud Computing, and Additive Manufacturing.

**Nuwan Ganganath** (S'09-M'17) received the B.Sc. (Hons) degree with first class honors in electronics and telecommunication engineering from the University of Moratuwa, Sri Lanka, in 2010, the M.Sc. degree in electrical engineering from the University of Calgary, Canada in 2013, and the Ph.D. degree in electronic and information engineering from the Hong Kong Polytechnic University, Hong Kong in 2016. He is currently an Adjunct Research Fellow at the School of Engineering at the University of Western Australia, Australia.



**Herbert Ho-Ching Iu** (S'98–M'00–SM'06) received the B.Eng. (Hons) degree in electrical and electronic engineering from the University of Hong Kong, Hong Kong, in 1997. He received the Ph.D. degree in Electronic and Information Engineering from the Hong Kong Polytechnic University, Hong Kong, in 2000.

In 2002, he joined the School of Electrical, Electronic and Computer Engineering, the University of Western Australia where he is currently a Professor. His research interests include power electronics, renewable energy, nonlinear dynamics, current sensing techniques, and memristive systems.

Prof. Iu currently serves as an Associate Editor of *IEEE Transactions on Power Electronics*, *IEEE Transactions on Smart Grids*, *IEEE Transactions on Network Science and Engineering*, *IEEE Transactions on Circuits and Systems–II* and *IEEE Access*.

**Chi K. Tse** (M'90–SM'97–F'06) received the BEng (Hons) degree in electrical engineering and the PhD degree from the University of Melbourne, Australia, in 1987 and 1991, respectively. He is presently Chair Professor at the Hong Kong Polytechnic University, Hong Kong, with which he was Head of the Department of Electronic and Information Engineering from 2005 to 2012. His research interests include power electronics, nonlinear circuits, and complex network applications.