OPTIMIZATION ALGORITHMS FOR ADDITIVE MANUFACTURING

KAI-YIN FOK

PhD

The Hong Kong Polytechnic University

2019

The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

Optimization Algorithms for Additive Manufacturing

Kai-Yin Fok

A thesis submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

May 2019

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

_____(Signed)

__Kai-Yin Fok__(Name of student)

Abstract

This thesis is dedicated to the study of 3D printing optimization algorithms. By minimizing the time needed by a printing nozzle to traverse a tool-path, the fabrication time of 3D objects can be shortened. Apart from fabrication time, other physical properties of fabricated parts such as dimensional accuracy and visual quality are considered as important evaluation criteria in this thesis. The tool-path planning problem of 3D printing applications is formulated into an undirected rural postman problem (URPP), which can be solved using existing URPP solvers. However, the computational times required by them increase rapidly with the number of print segments in a model. Some of these solvers are too time-consuming to be applied in real-life 3D printing processes. This thesis aims to develop algorithms to enhance the efficiency of 3D printing processes for both industrial and household applications.

This thesis begins with a detailed problem formulation, followed by an introduction to some essential tweaking and modification techniques, which include a tool-path optimizer developed based on Christofides' algorithm and a 2-opt local search algorithm. Furthermore, a segment-consolidation scheme is proposed to shorten the optimization process. The tool-path optimizer is operating at the inter-partitions level and the intra-partition level sequentially, with an aim to find a sequence with a low time-cost to visit all dissected parts on the same print layer of a model. Moreover, to represent curves on a layer, massive volumes of chained print segments were normally utilized. These print segments are consolidated

into replacement segments in the optimization process to reduce its computation complexity. The replacement segments are reverted to their corresponding print segments at the end of the optimization process. Simulation result using actual 3D models showed that the proposed tool-path optimizer can shorten the fabrication time of printed parts and required less post-processing time than its counterpart's.

Then, an ant colony optimization (ACO) based tool-path optimizer is proposed to further accelerate the printing processes. By utilizing the stochastic mechanism of the nature-inspired meta-heuristic and some unique properties in 3D printing processes, the proposed optimizer performs a more thorough search in the shrunk search space which helps to find better solutions when compared to generic ACO. Experiments were conducted using a domestic 3D printer and models to evaluate different tool-path optimizers. The proposed tool-path optimizer outperformed its counterparts in terms of both time-saving on fabrication times and post-processing times. Moreover, experiment results suggested that the proposed optimizer does not downgrade the dimensional accuracy of print parts but indeed improves the visual quality.

Finally, a computationally efficient tool-path optimizer is proposed based on a new detour search algorithm and an efficient local search algorithm. The new detour search algorithm focuses on replacing unnecessary overheads on a path with detours which have relatively lower time costs. While conventional k-opt local search algorithms are widely used to solve routing or path planning problems, their computational complexity increases rapidly with the parameter k. A new implementation is proposed based on some unique properties in 3D printing applications. The new implementation shrinks the searching space significantly. It is mathematically proven that the new implementation does not degrade the quality of the solutions generated. Both experiment and simulation results showed that the proposed tool-path optimizer can significantly accelerate the 3D print-

ing process by reducing at most one-third of the fabrication time when printing a model, which took insignificant post-processing time. The proposed tool-path optimizer is a local search algorithm, therefore, it is capable of cooperating with other algorithms, including the former mentioned two proposed optimizers, to further improve the quality of solutions.

Publications Generated From This Thesis

Journal papers

- **K. Y. Fok**, C. T. Cheng, N. Ganganath, H. H.-C. Iu, and C. K. Tse, "A nozzle path planner for 3D printing applications," Manuscript submitted for publication, 2019.

- **K. Y. Fok**, C. T. Cheng, N. Ganganath, H. H.-C. Iu, and C. K. Tse, "An ACO-based tool-path optimizer for 3-D printing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2277-2287, April 2019.

- C. T. Cheng, N. Ganganath, and **K. Y. Fok**, "Concurrent data collection trees for IoT applications," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 793-799, April 2017.

Conference papers

- **K. Y. Fok**, N. Ganganath, C. T. Cheng, H. H.-C. Iu, and C. K. Tse, "Tool-path optimization using neural networks," *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, Sapporo, 2019. Manuscript accepted for publication.

- **K. Y. Fok**, C. T. Cheng, N. Ganganath, H. H.-C. Iu, and C. K. Tse, "Accelerating 3D printing process using an extended ant colony optimization algorithm," *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, Florence, 2018, pp. 1-5.

- **K. Y. Fok**, C. T. Cheng and C. K. Tse, "A refinement process for nozzle path planning in 3D printing," *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, Baltimore, MD, 2017, pp. 1-4.

- **K. Y. Fok**, C. T. Cheng, C. K. Tse, and N. Ganganath, "A relaxation scheme for TSP-based 3D printing path optimizer," *2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, Chengdu, 2016, pp. 382-385.

- J. V. Wang, **K. Y. Fok**, C. T. Cheng, C. K. Tse, "A stable matching-based virtual machine allocation mechanism for cloud data centers," *2016 IEEE World Congress on Services (SERVICES)*, San Francisco, CA, 2016, pp. 103-106.

- **K. Y. Fok**, N. Ganganath, C. T. Cheng, and C. K. Tse, "A 3D printing path optimizer based on Christofides algorithm," *2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, Nantou, 2016, pp. 1-2.

- N. Ganganath, C. T. Cheng, **K. Y. Fok**, and C. K. Tse, "Trajectory planning for 3D printing: A revisit to traveling salesman problem", *2016 2nd International Conference on Control, Automation and Robotics (ICCAR)*, Hong Kong, 2016, pp. 287-290.

Acknowledgements

I would like to express my sincere gratitude to my supervisors Prof. Michael C. K. Tse and Dr. Chi Tsun Cheng for their continuous support and invaluable advice. They have broadened my horizons and provided me a precious opportunity. They inspired me and helped me at all times during my research. Never could I accomplish my study without their encouragement and inspirations.

I would also like to thank Prof. Herbert Ho Ching Iu, Dr. Nuwan Ganganath, and Dr. Jing Vicky Wang for many inspiring discussions and suggestions on my research. At the same time, I am grateful to all members in our Nonlinear Circuits and Systems Group for their help in the last four years.

The financial support provided by the Hong Kong Polytechnic University is gratefully acknowledged.

Last but not least, I would like to thank my parents for their love, care, and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Additive manufacturing is the technology that fabricates 3D models by depositing materials in a layer by layer manner. A typical example of additive manufacturing is fused deposition modelling (FDM) which is more widely known as 3D printing. To print a computer-aided design (CAD) model, the model is first broken down into multiple thin layers by using a slicer software. Among most off-the-shelf fused deposition modelling machines, their printing nozzles move along the print bed while printing, and the extruder controls the flow rate of the filament. Molten plastic filaments will be deposited via the printing nozzle onto the print bed to develop the print segments and construct the model layer by layer.

According to the report in [1], the global 3D printing market was valued at 4.4 billion U.S. dollars in 2013. The market size has been growing rapidly and it was valued at 12 billion in 2018. It is expected that the market size will reach 21 billion in 2021. It is reported that there are more than 1700 companies in the 3D printing business [2], which are mainly providing services on hardware, modelling, and printing. The most common type of material used in 3D printing applications is plastic, including PLA (Polylactic Acid) and ABS (Acrylonitrile Butadiene Styrene), which is suitable for a wide variety of applications due to its lightweight and low-cost [3]. Moreover, wood filament, which is a compound of

PLA and wood fibre, can be used to print parts with realistic wooden texture. The wood fibre can be made from recycled wood [4].

In addition, metal, including copper, steel, or titanium, has also been utilized in 3D printing as well, which enables rapid fabrication of metal parts with qualities close to those made using metal injection molding. Therefore, products with high resistance on temperature and pressure can be fabricated by using additive manufacturing technology. A 3D-printed titanium internal combustion engine was built in the University of Canterbury [5], which is demonstrated the capability of 3D printing in manufacturing applications. Recently, 3D-printed carbon-fibre bicycle frames [6] and 3d-printed airless bicycle tires [7] were unveiled. The time-frame of designing a bike frame, which was normally 18 months using the conventional method, can be reduced to 18 days by adopting the latest additive manufacturing technology [6]. These developments suggested that customized products including bikes can be fabricated within a local store with low cost, which could reduce the carbon footprint of shipping products which are normally more bulky than their corresponding raw materials.

Besides industrial applications, 3D bio-printing is becoming a hot topic nowadays which is another application of additive manufacturing. It enables 3D printing of biocompatible materials, cells, and supporting components into complex 3D functional living tissues [8]. 3D bio-printing has already been utilized for the generation and transplantation of, but not limited to, bone, heart tissues, and cartilaginous structures. The world-first surgery making use of additive manufacturing was conducted in 2017 [9]. The tibia bone of a man was removed due to infection. The 3D printed tibia transplant enables a new bone to grow around the outside of the 3D printed bone scaffold. The patient can walk once again after this revolutionary surgery. These cases indicated the importance of developing additive manufacturing technology.

1.1 Background

In a typical FDM based 3D printing process, a 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 an 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 PLA or 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. The 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 to stop extra filament from depositing 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.

1.2 Motivation

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*. According to our preliminary study, in general, around 36% of the whole print time is spent on moving the printing nozzle along non–print segments. 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, the visual quality is also regarded as an important criterion in 3D printing. One of the major causes of 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, the excess filament could leak from the nozzle unintentionally. Therefore, strings could be formed in these regions 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 experimental results in Fig. 1.1(c), such method alone is insufficient in tackling the strings issue.

Results shown in Fig. 1.1(c) were obtained using an ordinary domestic 3D printer [10] and a popular slicer software [11]. The printer is carefully calibrated to yield high structural accuracy and surface texture resolution (see. Fig. 1.1(b)). However, as shown in Fig. 1.1(c), traces of strings can still be observed even when retraction is enforced. Excessive strings appear between two disjoint parts on the same layer indicate there are rooms for further optimizations.

1.3 Problem Formulation

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.

(a) CAD model (b) Printed model (c) Strings

Figure 1.1: (a) CAD of the model "dragon_65_tilted_large" [12]. (b) The model printed together with the supporting structure. (c) Strings found on the printed model.

The nozzle path planning 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.

The optimization problem is a nozzle path optimization with additional constraints. It can indeed be considered as an alternated version of undirected rural postman problem (URPP) with constraints. URPP was first introduced by Orloff in [13]. The objective of this mathematical problem is to find a minimum-cost tour, in terms of time or distance, which traverses a set of required segments at least once. URPP is proven to be NP-hard if the set of required segments is not equal to the complete segment set [14]. 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 its 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 [15] are elaborated as follows. 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

$$\sum_{e \in E_D} c_e x_e, \tag{1.1}$$

subject to:

$$\sum_{e \in \Omega_D(v)} x_e \ \equiv\ |E_r(v)| \bmod 2,\ v \in V, \tag{1.2}$$

$$\sum_{e \in \Omega_D(S)} x_e \ \geq\ 2,\ S \subset V,\ \Omega_r(S) = \phi, \tag{1.3}$$

$$x_e \in \mathbb{Z} : x_e \geq 0,\ e \in E_d. \tag{1.4}$$

Here, constraint (1.2) establishes the Euler property of $E_r \cup E_p$ while connectivity is guaranteed by constraint (1.3).

In this thesis, the nozzle path planning problem is transformed into a URPP with additional constraints. The nozzle path planning 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_{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_{r} and an extra vertex v_{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

$$|V| = 2n + 1. \tag{1.5}$$

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

$$t(v_i, v_j) = t_{\mathrm{d}}(v_i, v_j) + t_{\mathrm{r}}(v_i, v_j). \tag{1.6}$$

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

$t(v_i, v_j) = t(v_j, v_i)$, $\forall v_i, v_j \in V$.

The motion model adopted in this thesis is presented as follows. 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 $v_{\max}$.

In order to minimize the time cost, the nozzle accelerates 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

$$d_{\min} = \frac{v_{\max}^2}{2}\left(\frac{1}{a_1} - \frac{1}{a_2}\right). \tag{1.7}$$

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

$$t_d = \begin{cases} \sqrt{2d\left(\frac{1}{a_1} - \frac{1}{a_2}\right)} & \text{if } d \leq d_{\min}, \\ \frac{v_{\max}}{a_1} - \frac{v_{\max}}{a_2} + \frac{d - d_{\min}}{v_{\max}} & \text{otherwise.} \end{cases} \tag{1.8}$$

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

A feasible solution to a nozzle path planning problem is a directed path that leads the printing nozzle through all required print segments at least once starting from a predefined starting vertex. Meanwhile, the ending vertex is not necessary

to be equivalent to the starting vertex. The printing nozzle does not return to its starting location. The printing nozzle stops or moves to the next layer right after it traverses all required print segments on the current layer.

1.4 Thesis Organization

This thesis is organized as follows. Chapter 2 provides a literature review. Key researches on improving additive manufacturing in variety of aspects, algorithms on solving combinatorial optimization problems in industrial applications, and the algorithms focusing on path planning problems are reviewed.

Chapter 3 discusses a fundamental framework for optimizing the nozzle path planning problem. Based on a Christofides' algorithm, some unique characteristics of the nozzle path planning problem, and combined with a segment-consolidation scheme, a nozzle path optimizer is introduced to shorten the transition length of a printing nozzle. To evaluate the performance of the optimizer, simulations were conducted. The performances of different optimizers are compared and discussed.

Chapter 4 proposes an ant colony optimization (ACO) based optimizer. ACO is utilized as a path planner to generate fast print plans for 3D printers. By performing careful tool–path optimization, the printing process can be speeded up, while the visual quality of printed objects can be improved simultaneously. Moreover, modifications on the ACO have been made which exploits unique properties in 3D printing to further improve the solution quality and shorten the computational time at the same time. Experiments and simulations were conducted to evaluate the performance of the proposed method.

Chapter 5 proposes a new computationally efficient heuristic search algorithm. Typical local search algorithms often require lengthy computational times to find fast and feasible routes for printing meticulous products. Moreover, the compu-

tational times required by these methods can increase rapidly with the number of printing segments. A new local search algorithm is proposed which shrinks the searching space without degrading the quality of the solution. Experiments and simulations showed that the proposed method can find fast routes and mitigate overheads in printing 3D printing applications.

The thesis concludes in Chapter 6, where major findings of the project are summarized and some thoughts on the future works are presented.

Chapter 2

Literature Review

Improvements and breakthroughs in additive manufacturing technology have been largely reported in recent years. In this chapter, different approaches applied to improve additive manufacturing processes are reviewed. Furthermore, algorithms for solving tool-path planning problems are chosen to discuss further.

2.1 Different Optimization and Enhancement Approaches

2.1.1 Printing Orientation

Printing orientation of a 3D model can significantly affect its structural stability and the amount of support materials needed. Son and Choi studied the effects of different 3D model printing orientations to three printing performance indices [16], namely a) amount of support material used, b) resolution error on the Z dimension, and c) visual artifacts due to support residue. Based on these performance indices, they proposed an automated orientation selection method, which can yield higher efficiency and accuracy in 3D printing processes. Accuracy and surface finishing of printed objects are important criteria in additive manufac-

turing. In [17], Armillotta studied the surface quality of textured printed objects and provided guidelines on selecting suitable object scale and orientation without compromising the texture quality. Cheng *et al.* in [18] studied how fabrication accuracy, print time, and cost of an FDM process be affected by the orientation of the printing model. They showed that by optimizing the orientation of a model, the amount of supporting materials needed can be greatly reduced and thus can shorten the print time.

2.1.2 Tool-path Generation

Apart from printing orientation, filling patterns are crucial factors 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 print time. Zarbakhsh *et al.* in [19] considered about the physical properties of printed objects. They analyzed regions of interest on printed objects by adopting a technique called sub-modeling. Their results show that object filling patterns can introduce extra stress between printed layers. Agarwala *et al.* [20] further considered the precision and the physical properties of the internal structure in fabricating ceramic and metal components. A similar study was conducted by Lim *et al.* [21] 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.* [22] demonstrated an improved printing quality and accuracy by generating parallel infilling segments with adaptive separations. 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 [23] and [24], respectively.

Wang *et al.* [25] studied the visual quality of printed objects. They proposed an adaptive slicing scheme which is capable of selecting different layer thicknesses

according to the structures of 3D models. They showed that the corresponding print time can be reduced by using their method while high visual quality can still be preserved. Ezair *et al.* [26] studied the generation of support structures in 3D printing. In order to minimize the amount of materials used to form the support structures, they proposed an algorithm which can select an appropriate orientation according to the model to be built.

2.1.3 Tool-path Planning

Recently, research works [27–32] have demonstrated that the total length of transitions can be reduced by attentively planning the printing sequence and thus allowing the print time to be shortened. Thompson and Yoon [27] developed a path planning algorithm to minimize the amount of material wasted during aerosol printing processes. The time spent on the transitioning between print segments can be reduced by using two motion control methods proposed, including linear segments with parabolic blends and minimum time trajectory. Freens *et al.* [28] 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.

Some algorithms designed for travelling salesman problem (TSP) can also be applied to solve tool-path planning problem [29, 30]. TSP is a well-known combinatorial optimization problem which aims to search the shortest tour that visits every node exactly once and return to the starting node. The major difference is that the former focuses on connecting existing edges while the latter focuses on joining nodes. TSP is proved to be MAX SNP-hard [33]. In [29], Christofides' algorithm was used to perform robot path planning in an automotive manufacturing application. A similar problem about generating a route for a printing

nozzle of an FDM printer was also formulated into TSP in [30], where a genetic algorithm (GA) was employed for solving the underlying large scale TSPs within reasonable durations [30]. The total transition time can be further reduced by appending a 2-opt or 3-opt local search algorithm to the end of their proposed method. They showed that local search processes using 3-opt are significantly longer than those with 2-opt. Besides, it is found that integer programming took even longer processing time and sometimes failed to generate solutions within a reasonable time frame. The tool-path planning problem for spray forming processes was formulated into URPP in [32]. In [31], Tewolde and Sheng evaluated the performance of different meta-heuristics including ACO and GA in solving robot tool-path integration problems. More details on meta-heuristics will be discussed in the next section.

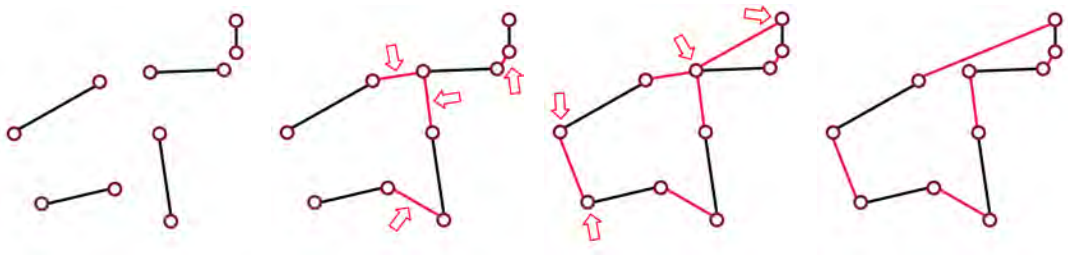
2.2 Algorithms on Tool-path Planning

In the last section, it has been shown that tool-path planning problems can be formulated into TSP or URPP. In this section, well-known solvers for these problems are introduced.

2.2.1 Christofides' and Frederickson's Algorithm

Christofides proposed a polynomial time algorithm for solving TSP in [34]. This deterministic algorithm is proven to have an approximation factor of 1.5 in solving TSP [34]. Later, Frederickson's algorithm [35] 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 tree (MST), performing a minimum cost perfect matching, and shortcuts searching.

The procedures of the Frederickson's algorithm are shown with the aid of Fig 2.2 as follow.



(a) (b) (c) (d)

Figure 2.1: Illustrations of the work-flow of the Frederickson's algorithm. Black and red lines represent edges and transitions, respectively. The dots represent vertices.

1. Frederickson's algorithm begins with a given undirected and connected graph $G = (V, E)$ and a set of required edges $E_{\mathrm{r}} \in E$, as shown in Fig 2.1(a).

2. A MST is then constructed to connect all the edges in E_{r}. Those new edges introduced in the MST construction process form a new set E_{mst}. The edges in E_{mst} are indicated in Fig 2.1(b).

3. A minimum cost perfect matching is then conducted on the sumset $E_{\mathrm{r}} + E_{\mathrm{mst}}$ to connect vertices with odd degrees. The nodes with odd degree are indicated in Fig 2.1(c).

   Those extra edges added in the matching process form another new set E_{matching}. An Eulerian tour can then be found in the sumset $E_{\mathrm{r}} + E_{\mathrm{mst}} + E_{\mathrm{matching}}$.

4. Consecutive edge pairs on the tour, whose edges are not in E_r, are replaced with shortcuts to further optimize the tour. A tour can be obtained as shown in Fig 2.1(d).

2.2.2 Heuristics and Meta-heuristics

In addition to the tool-path planning problem discussed in the last section, meta-heuristics were also widely adopted in solving combinational optimization problems [36–43] Cheng *et al.* [36] proposed a GA-inspired unmanned underwater vehicles path planner based on dynamic programming, which can solve large-scale path planning problems within reasonable durations. Delaram *et al.* considered the thermal placement optimization problem in printed circuit boards. Their work [37] showed that the optimization processes of thermal chip placement can be improved with GA. Their meta-heuristic approach can generate circuit designs with the same quality in terms of their reliability and modules performance with shorter processing time. Yahyaoui *et al.* [38] 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.* [39] considered the stack palletizer scheduling problem, which is NP-hard. In their work, a GA-based solver was proposed. In [40], Lee *et al.* proposed an ant colony optimization (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 [41]. Xu *et al.* [42] developed an automated assembly planning system, where the critical challenge goes to the complexity of the planning problem itself. The complexity of the problem is usually high, and it increases rapidly with the number of parts to be assembled. To obtain a fast assembly plan, the authors utilized several meta-heuristics, including GA and ACO. Alkaya and Duman [43] aimed to increase the efficiency of an automated assembly process of printed circuit boards (PCBs). The objective is to find a component placement sequence that can minimize the assembly time. Promising results were obtained by utilizing a simulated annealing (SA) followed by a local search algorithm designed for TSP.

Researches [44,45] have been conducted to analyze the performance of different meta-heuristics which are frequently used in solving TSP. The results suggested that, in general, solvers based on ACO can deliver more 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. Therefore, a detailed discussion on ACO is given below.

ACO is a meta-heuristic which is inspired by the collective behaviours of ants. It was first proposed to solve TSP [46]. ACO can also be used to solve variations of TSP, such as a dynamic travelling salesman problem (DTSP) [47]. When ACO is applied to solve 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. The 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. Given that the k-th ant is currently at vertex i while constructing its path. The probability of choosing a path (i, j) is expressed as

$$p_{i,j}^{k}(t) = \frac{[\tau_{i,j}(t)]^{\alpha}[\eta_{i,j}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{i,l}(t)]^{\alpha}[\eta_{i,l}]^{\beta}}. \tag{2.1}$$

The heuristic value between the 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_i^k 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 α and β are control parameters used in generic ACO that can affect the quality of solutions

generated. Further information on α and β can be found in [46]. 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

$$\tau_{i,j}(t+1) = (1-\rho)\tau_{i,j}(t) + \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}(t). \tag{2.2}$$

Here, parameter ρ 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 $\Delta\tau_{i,j}^{k}(t)$. The value of $\Delta\tau_{i,j}^{k}(t)$ 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. Ultimately, majority of the colony will therefore move along high quality path segments that connect the starting and the ending points. Extensive researches have been conducted to improve the performance of ACO. Zheng *et al.* [48] proposed a parameter-adaptive strategy for ACO. In their work, the parameters can be adjusted dynamically during the optimization process. Mahia *et al.* [49] utilized a PSO method to optimize the parameters in ACO according to the quality of solutions generated by the ants. Skinderowicz [50] proposed several parallel versions of ACO which utilized the graphics processing units (GPUs).

2.2.3 Local Search Algorithms

In this section, the k-opt local search algorithm is introduced, which can be used a refinement process for optimizing TSP and URPP solutions. k-opt algorithm was first designed to further optimize sub-optimal TSP solutions [51]. It was extended by Hertz in [52] to support URPP. In [53], Muyldermans *et al.* utilized a k-opt local search algorithm in solving general routing problems (GRP).

(a) (b) (c) (d) (e)

Figure 2.2: Illustrations of the work-flow of the 2-opt local search algorithm.

The procedures of a conventional k–opt algorithm are shown with the aid of Fig 2.2 as follows.

1. The algorithm starts with an input tour as shown in Fig 2.2(a).

2. At the beginning of each iteration, the algorithm evaluates all combinations one by one. A combination refers to the way of breaking down a tour into k fragments and reconnecting them. Two highlighted edges are selected in Fig 2.2(b).

3. After checking all available combinations, the tour is updated according to the combination that associated with the best improvement. The two highlighted in Fig 2.2(b) are replaced with two new edges in Fig 2.2(c).

4. The algorithm returns to step 2 with the updated tour. Another two edges in Fig 2.2(d) are placed with new edges as shown in Fig 2.2(e). The algorithm returns a tour if no more improvement can be found.

Similar to [53], when a k-opt algorithm is used to solve a URPP with n required edges, the total number of possible combinations to be evaluated in the search space in each iteration is

$$\frac{n!\ 2^{k-1}}{k!\ (n-k)!}. \tag{2.3}$$

The total number of possible combinations increases exponentially with k and n. Previous studies [53] showed that 2-opt and 3-opt algorithms can deliver

promising results. However, 3-opt algorithm usually requires significantly longer processing times, which may not be applicable in some scenarios. The k-opt local search algorithm was often applied to further optimize solutions generated by other algorithms or meta-heuristics. It was shown that better solutions to RPP can be obtained by applying a 2-opt or 3-opt local search algorithm to solutions generated by Frederickson's algorithm [54]. Gülcü *et al.* [55] proposed a hybrid method based on an ACO and a 3-opt local search algorithm for solving TSP. Desirable results were obtained, however, with a high computational cost.

2.2.4 Neural Networks

Machine learning is a promising tool for solving problems including classification problems and combinatorial optimization problems. Neural networks (NN) have demonstrated their capability in solving pattern recognition problems and have been applied in real-life applications. Recently, a human action recognition system using an NN was proposed [56]. Meta-heuristics were utilized to minimize its classification error. It showed that with a proper feature extraction, NNs are capable of identifying complicated actions. In [57], Zhang *et al.* developed a fuzzy energy management controller using a NN for identifying the driving pattern of vehicles with fuel cells. The outcomes can minimize the energy consumption of the vehicles and prolong the lifetime of the fuel cells. In [58], a NN was utilized to solve a TSP. NNs have also demonstrated promising performances in generating results close to optimal results on solving small TSP instances. The method proposed in [58] can successfully find optimal results up to 100 nodes. At the same time, authors of [58] have raised concerns over its processing time. The above studies showed some insights in applying machine learning approaches to solve the tool-path planning problem in additive manufacturing applications.

In this chapter, a literature review on the improvements made to additive

manufacturing in recent years has been provided. It is observed that most proposed approaches were focusing on printing orientations, tool-path generation, and tool-path planning. Majority of the works suggested that by performing careful tool-path optimization, the fabrication time can be reduced and the visual quality can still be preserved. Thus, in the following chapters, the nozzle path planning problem in additive manufacturing is firstly formulated into a URPP and then solved by our proposed algorithms. The performances of the proposed algorithms were evaluated based on the solution quality, processing time, model print time, and model visual quality.

Chapter 3

Segment-consolidation Scheme

In the previous chapter, recent studies on TSP and URPP are discussed. In this chapter, a too-path optimizer and a segment-consolidation scheme are proposed, which exploit the unique properties in 3D printing processes.

3.1 Introduction

Researchers [27–32] have reduced the fabrication time of different industrial applications by shortening their overheads by using proper formulations and modification. Solvers for TSP, URPP and other graph search problems were utilized in these studies.

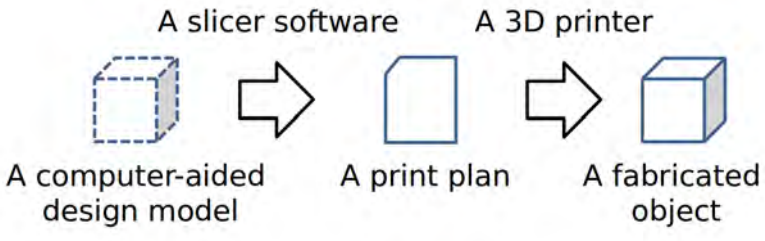


Figure 3.1: A standard procedure of 3D printing processes

Fig. 3.1 shows the standard procedures in a 3D printing process. A 3D printing process starts with a CAD model as shown in Fig. 3.3(a). For typical 3D printers,

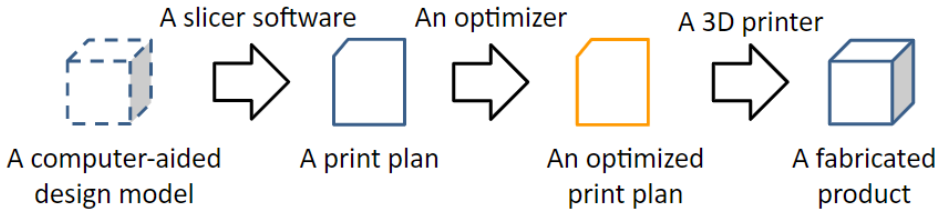
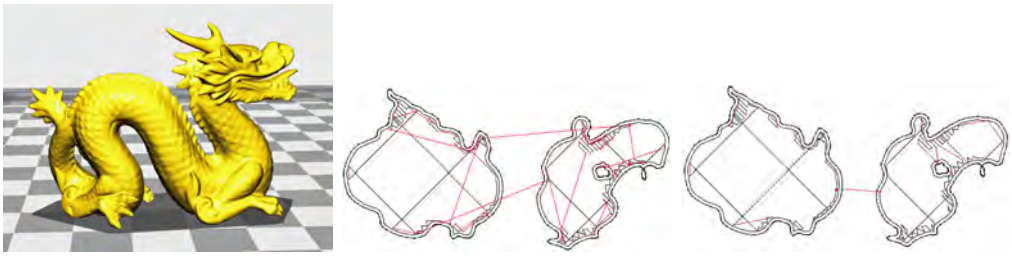


Figure 3.2: The procedure with a tool-path optimizer

in order to build an object, a CAD model is first sliced into many thin layers. Object fragments in each layer are then constructed by a massive volume of segments. A segment is a straight line defined by its two endpoints. Curves are represented by multiple segments. Segments are further categorized into two types. The first type is *print segment*, which represents the print part of the model. The printing nozzle will deposit filament when traversing through a print segment. In order to maintain consistent material deposition, the rate of depositing is adjusted jointly by the speed of the extruder wheel and the motion of the printing nozzle. The second type of segments is *transition*, which represents the paths for the nozzle to traverse between print segments. As an example, here, a print plan is generated from its CAD using a slicer software. The print plan of the 46th layer of the model is shown in Fig. 3.3(b), black and red lines represent print segments and transitions, respectively. The print plan is then sent to a 3D printer for printing.



(a) (b) (c)

Figure 3.3: (a) CAD of the model "dragon_65_tilted_large" [12]. (b) The print plan of the 46th layer. Black and red lines represent print segments and transitions, respectively. (c) An optimized print plan.

The objective of this work is to develop an optimizer to accelerate the 3D printing process. In this chapter, a tool-path optimizer based on the problem formulation introduced in Section 1.3 is proposed to accelerate the 3D printing processes. To reduce the computational complexity of the nozzle path planning process, a segment-consolidation scheme is proposed. The proposed scheme contains two major steps, namely partitioning and consecutive segments consolidation. Section 3.3 shows and discusses the results of the simulations using the method proposed. A concluding summary is given in Section 3.4.

3.2 Tool-path Optimizer

In this chapter, the nozzle path planning problem in each layer is formulated into a URPP as in Section 1.3. The main objective of the optimizer in this chapter is to reduce the print time required by the printing nozzle to traverse all the transitions. A tool-path optimizer is proposed to shorten the print time. Here, *print time* denotes the time required by a 3D printer to fabricate an object. After a print plan is generated by a slicer software, the proposed optimizer is utilized to further optimize the print plan (as shown in Fig. 3.2). An illustration of a layer of an optimized print plan is shown in Fig. 3.3(c). With further optimization, the time required by the printing nozzle to traverse the transitions, which are represented by red lines, is shortened as the total transition length has been reduced. The 3D printing process then proceeds with the modified print plan. Here, the time required by the optimizer in the tool-path planning process is denoted as *post-processing time*. It is worth to note that in this work, an optimizer is considered as impractical if post-processing time required by the optimizer is longer than the time-saving in print time obtained by the optimizer.

A preliminary study was conducted to examine whether the 3D printing process can be accelerated by conducting tool-path planning. A Christofides' algo-

rithm and a 2-opt local search algorithm introduced in the Sections 2.2.1 and 2.2.3 were utilized together as a generic tool-path optimizer for benchmarking purposes. The preliminary results showed that the optimizer can reduce the total transition length with an expense of a long post-processing time. Details of the preliminary study will be given in Section 3.3. In order to develop a practical optimizer to accelerate 3D printing applications, a segment-consolidation scheme is proposed to reduce the computational complexity of the nozzle path planning process.

3.2.1 Partitioning

The first step of the scheme is to partition a layer of the print plan into sub-parts. By doing so, nozzle path planning processes can be performed on those smaller parts separately which are in general associated with lower computational complexities. As a 3D object is printed layer-by-layer, although the object may not have separated components, after slicing, there can be discrete print parts on some of the layers as shown in Fig. 3.3(b). In this thesis, the slicing process which transforms a computer-aided designed model into a print plan is considered as out of scope. A print layer can therefore be partitioned according to its boundaries. Print segments located inside the same boundary are assigned with the same partition index. the nozzle path planning will then be carried out at the inter-partitions level and the intra-partition level sequentially. At the inter-partitions level, centroids of the partitions are regarded as the vertices to be visited. Here, partitions are represented as vertices. Each vertex is the centroid of a partition, which is the average point of all the points located inside that partition. Details will be described in the next section.

3.2.2 Consecutive Segments Consolidation

As mentioned in Section 3.1, curves on a print layer are represented by chains of connected short print segments. The computational complexity of the nozzle plan planning will be largely increased if we consider those segments as separated components. On the other hand, the complexity of the problem can be significantly reduced by consolidating multiple connected print segments into a single replacement segment during calculations. Here, a replacement segment represents a chain of print segments in the print plan is used to replace the chain of print segments during the optimization process in order to reduce the computational complexity of the problem. Note that such process needs to be carried out with cautions, as consolidating a very long chain of print segments may eliminate branching options at the middle of the chain which may yield better results in the nozzle path planning. A consolidation threshold ϕ is therefore introduced here, which controls the maximum length of print segments to be consolidated at a time. Effects on the selection of ϕ are further discussed in Section 3.3. At the end of the nozzle path planning, replacement segments are reverted to the corresponding chains of print segments.

Given the print segments on the current layer, the optimization process will begin with the partitioning task. A nozzle path planning process will be performed to find a fast tour to visit all the centroids of the identified partitions, which also determines their order of visit. Inside each partition, the Christofides-based TSP solver with 2-opt in [59] is used to find a fast tour. To shorten the transition segments among partitions, the virtual segment required by the method in [59] is placed between two end points that are nearest to the centroids of the adjacent partitions. This will ensure the tour will always start and end at these two selected points. An extra transition segment will be added to guide the printing nozzle to the nearest selected end point mentioned above. Transition

segments will then be added to join those selected points of adjacent partitions together and form a single tour.

3.3 Simulations

In this section, performances of the proposed segment-consolidation scheme and the proposed tool-path optimizer introduced in the last section were evaluated using computer simulations. Simulation parameters and results are presented in the later parts of this section.

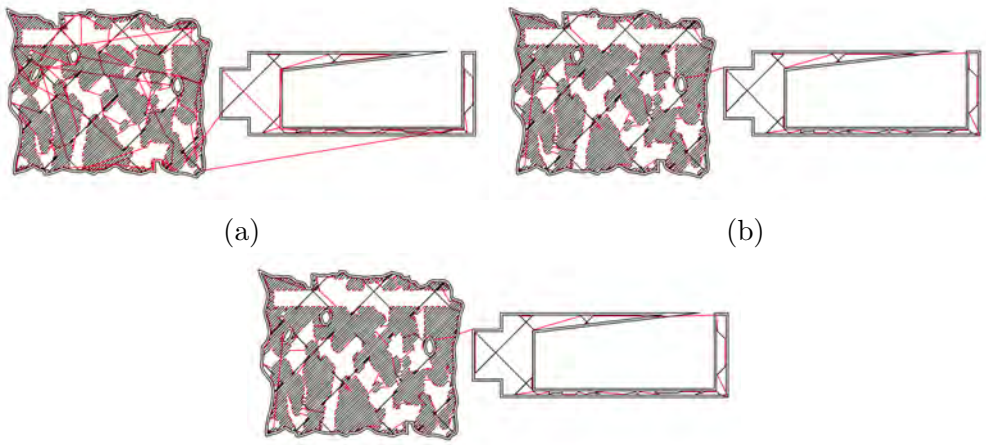
3.3.1 Simulations Settings

In the simulations, seven 3D models in [60] were chosen as sample models. The sizes of the models were adjusted such that they can fit in the print platform of a typical 3D printer, which is 250×250 mm^2. The consolidation threshold ϕ used in the proposed scheme is 4 mm which is arbitrarily selected. The print segments of each model were generated using Cura [61]. The order for visiting those print segments were further optimized using the proposed scheme and the optimizer mentioned in Section 3.2. In Cura, the speed for traversing a print segment is set to 50 mms^{-1} while that for traversing a transition segment (i.e. $v_{\max}$) is set to 150 mms^{-1}. Retractions were enabled with a retraction length of 4.5 mm. Vertical hop during retracting was set to 0.075 mm. The filling density was 10% of the total volume. All print plans were evaluated using the Code Print Simulator-1.32 [62]. The processing time required by the proposed method are the average values obtained from 10 individual simulations conducted on a computer with Windows 8.1, Intel Core i7 processor, and 16 GB RAM. All the results are presented in Table 3.1.

3.3.2 Results and Discussion

Table 3.1: Print time and post-processing time of different tool-path optimizer under test.

3D models	Built-in algorithm	Proposed tool-path optimizer			
		Without segment-consolidation scheme		With segment-consolidation scheme	
	Print time (s)	Print time (s)	Post-processing time (s)	Print time (s)	Post-processing time (s)
UltimakerRobot_support_2015	1640.05	1590.08	44.92	1593.69	5.93
TortureTestV2	7556.58	6886.78	71.84	6775.95	57.36
testModel	1688.88	1579.4	3.06	1584.68	1.37
dragon_65_tilted_large	2499.21	2070.62	114.81	2093.85	10.71
Debailey_x10	3735.86	3306.85	38.19	3313.21	9.3
ctrlV_3D_test	3393.68	2955.31	201.82	2967.41	83.86
3DHackerTest	4566.3	3623.69	590.03	3881.92	123



(a) (b)

(c)

Figure 3.4: Illustrations of print plans obtained using (a) the built-in path optimizer in Cura, (b) the proposed tool-path optimizer without segment-consolidation scheme, and (c) the proposed tool-path optimizer with segment-consolidation scheme. Transitions and print segments are presented by red and black lines, respectively.

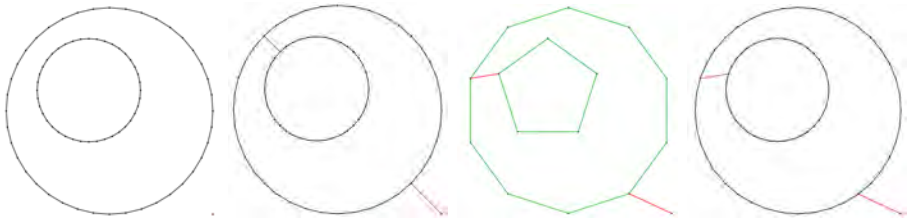
According to the simulation results given in Table 3.1, it can be observed that the proposed tool-path optimizer without segment-consolidation scheme can generate print plans that required 0.88% shorter print time to those of the tool-path optimizer with the segment-consolidation scheme. Figs. 3.4(a), 3.4(b), and 3.4(c) illustrate the print plans of the 60th layer of the model "3DHackerTest" obtained by Cura, the proposed optimizer without segment-consolidation scheme, and the proposed optimizer with segments consolidation scheme, respectively. It

can be observed that the total length of transitions can be greatly reduced after using the proposed optimizer. At the same time, the print plans in Fig. 3.4(b) and Fig. 3.4(c) are very similar even though a segment-consolidation scheme was utilized on one of them. However, the tool-path optimizer took 265% longer post-processing time without the help of the segment-consolidation scheme. As a result, by comparing the average total print times (print time + processing time), the tool-path optimizer with segment-consolidation scheme can generate print plan with on average 2.35% shorter total print time than that of the tool-path optimizer without the segment-consolidation scheme.

According to the simulation results given in Table 3.1, it can be observed that the proposed method can on average reduce object print time by 10.63% when comparing with Cura. Even by including the processing time of the proposed method, while assuming such time of Cura is negligible, an average total saving in print time (print time + processing time) of 9.62% can still be achieved.

It is worth to note that for the results associated with the model "Ultimaker-Robot_support_2015", the proposed method can only deliver a saving of 2.46% on the total print time over Cura. One possible reason is due to the compact structure of the model, where most print segments are connected together and have left very limited rooms for optimization to take place. In contrast, the proposed method can achieve a saving of 11.06% on the total print time over Cura on the model "dragon_65_tilted large", which consists of a large number of separated components.

As mentioned in Section 3.1, consecutive print segments are substituted by replacement segments in the proposed scheme depending on the consolidation threshold. The selection of such threshold can be crucial to the solution quality. A small value of ϕ will not have significant effects on reducing processing time as very few print segments can be consolidated. A large value of ϕ can greatly reduce the complexity of the problem but may reduce branching options in the tool-



(a) (b) (c) (d)

Figure 3.5: Illustrations of print plan using different values of ϕ. (a) Print segments (b) Result with ϕ equals 0 (c) Replacement segments with ϕ equals 70 mm (d) Result with ϕ equals 70 mm. Transitions, print, and replacement segments are shown in red, black, and green colour, respectively.

path planning process and end up with longer tours. Fig. 3.5 shows a simplified example illustrating how such threshold may affect the print time.

Consider a set of print segments as shown in Fig. 3.5(a), which appear as two tetracontagons. Suppose the printing nozzle is located at the bottom right corner. When ϕ is smaller than the length of the shortest segment, no segment-consolidation will be executed. A nozzle path planning process will be performed by considering them as 80 separated segments and yield the result shown in Fig. 3.5(b). For a larger value of ϕ, multiple consecutive segments can be substituted by replacement segments in the calculation process. In this example, the computational complexity can be greatly reduced as the number of segments is reduced from 80 to 15 as shown in Fig. 3.5(c). However, it can also be observed that the length of transition segments in Fig. 3.5(d) is slightly longer than that in Fig. 3.5(b). In general, for most scenarios, selecting a large value of ϕ is recommended as the negative impact due to longer transition paths is not significant to the total print time which is often eliminated by the huge reduction in processing time.

Note that during the tool-path planning processes, to ensure the integrity of the printed model, none of the original print segments is reoriented or skipped. Therefore, accuracy and the physical properties of the printed model should not be affected by the proposed optimizer in general. However, it is worth to in-

vestigate whether the selection of transitions, which are generated based on the traversing sequence of the print segments, would affect the accuracy of the printed model. Experiments and discussions are represented in the next chapter.

3.4 Summary

In this chapter, a tool-path optimizer and a segment-consolidation scheme for nozzle path planning problem in 3D printing applications are proposed. This chapter proposes essential modifications to typical URPP-solvers for optimizing tool-paths in 3D printing. A tool-path optimizer was developed based on a Christofides' algorithm and a 2-opt local search algorithm. The segments consolidation scheme reduces the computational complexity of the tool-path planning problem in 3D printing by partitioning each layer of the print area into sub-parts and consolidating consecutive print segments in the calculations. Performance of the proposed scheme is evaluated using computer simulations. Simulation results show that the proposed optimizer can significantly reduce the print time comparing to a tool-path optimizer used in a common 3D model slicing software. Simulation results also show that the proposed scheme can greatly shorten the processing time of tool-path optimizer without introducing significant impacts on the solution quality.

Chapter 4

ACO-based Tool-path Optimizer

In the previous chapter, a tool-path optimizer and a segment-consolidation scheme for the nozzle path planning problem are proposed and evaluated. In this chapter, a new tool-path optimizer based on ACO is proposed, which exploits unique properties in 3D printing.

4.1 Introduction

Throughout the years, meta-heuristics have been widely applied to improve the performance of industrial applications, including a job-shop scheduling problem [38], a stack palletizer scheduling problem [39], a sensing schedule problem in wireless sensor networks [40], robot motion control [41], and tool-path integration problems [31]. To accelerate the 3D printing processes, a new optimizer based on ACO is proposed. The proposed method further extends the concept of pheromone in ACO to shorten optimization processes.

During a 3D printing process, most of its print time is spent on moving the printing nozzle along print segments and transitions. Similar to the previous chapter, this chapter also focuses on accelerating 3D printing processes by shortening the time spending on transitions, which 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 constrained by the physical properties of the printing material and the printing machine, the total print time can be shortened by having shorter transitions.

Apart from object print time, the visual quality is also an important criterion in 3D printing. One of the major causes of 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, the 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 experimental results in Fig. 4.1(c), such method alone is insufficient in tackling the strings issue.

Results shown in Fig. 4.1(c) were obtained using an ordinary domestic 3D printer [63] and a popular slicer software [61]. The printer is calibrated to yield high structural accuracy and surface texture resolution (see. Fig. 4.1(b)). However, as shown in Fig. 4.1(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 chapter, an ACO based tool-path optimizer for 3D printing applications is proposed to shorten printing processes and improve the visual quality of printed models simultaneously. The proposed optimizer accelerates printing processes by eliminating unnecessary movements of the printing nozzle and alleviates the

strings issue in 3D printing by discouraging the nozzle to hop across boundaries of disjoint parts.

The proposed tool-path optimizer is introduced in Section 4.2, which exploits the unique characteristics in 3D printing. The modifications made to ACO are elaborated in the same section. The proposed optimizer was evaluated using both simulations and actual 3D printing experiments. Results are presented in Section 4.3. The results are analysed and discussed in Section 4.4. Concluding summary are given in Section 4.5.

4.2 Proposed Tool-path Optimizer

In this section, a two-stage optimizer is proposed to solve the tool-path optimization problem in additive manufacturing. The proposed optimizer utilizes 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 generated by a slicer software. As mentioned in Section 4.1, in this work, the process for

(a) (b) (c)

Figure 4.1: (a) CAD of the model "dragon_65_tilted_large" [64]. (b) The model printed together with the supporting structure. (c) Strings found on the printed model.

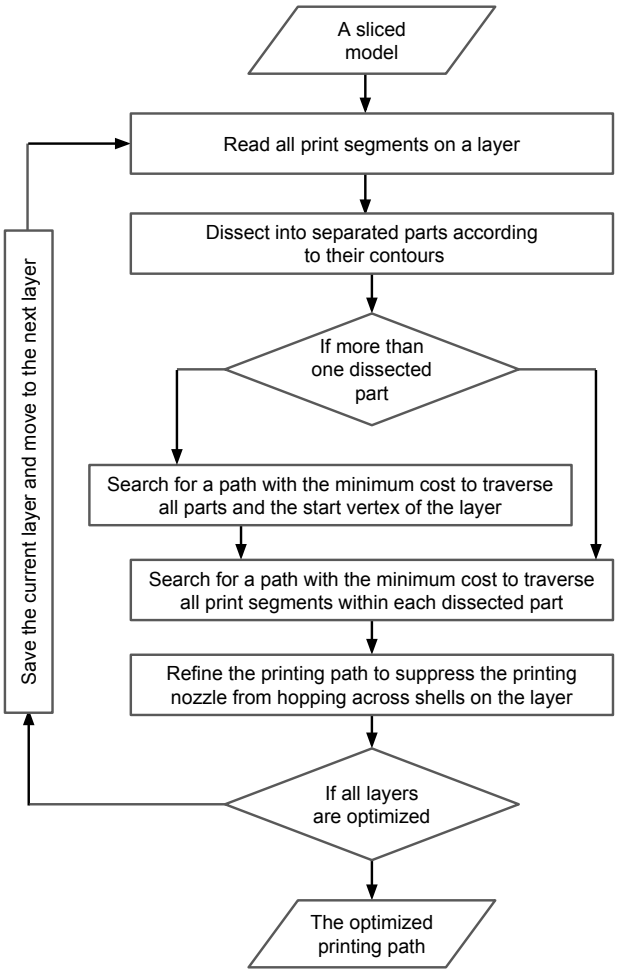


Figure 4.2: The workflow of the proposed optimizer.

generating print segments from the 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 are not affected by the proposed optimizer in general.

The proposed optimizer has a two-layer architecture, which is illustrated in Fig. 4.2. 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.

4.2.1 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.

After finding a fast path visiting sequence, all disjoint parts are connected with a route. An illustrative example is shown in Fig. 4.3, 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.

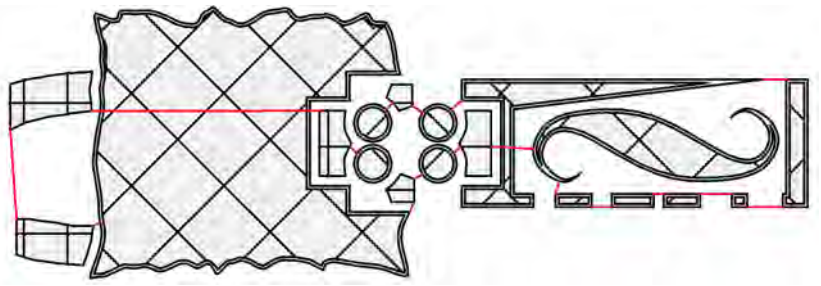


Figure 4.3: An illustration of a layer of the model "3DHackerTest" [64]. The black lines represents the print segments. Interiors of each part is shaded in grey color. The red lines represent the transitions that connect disjoint parts.

4.2.2 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 proposed ACO-based URPP solver and other solvers under test will be introduced and explained in the next section. After obtaining a tour that traverses 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.

4.2.3 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 [65] can be used to achieve such a goal.

With the results obtained from the process mentioned in Sections 4.2.1, 4.2.2, and 4.2.3, printing paths of disjoint parts on the current layer can be integrated into a single route.

4.2.4 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 a 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 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

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

Here, the solution S_a represents a route for the printing nozzle. S_a contains $2n$ points, which contains n transitions to connect all n print segments to v_{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, $(v_{a_{2i}}, v_{a_{2i+1}})$ 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 α and β 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

$$p_{(i,i+1)}(t) = \min\left(\frac{\theta n[\tau_{2i,2i+1}(t)]^{\alpha}[\eta_{2i,2i+1}]^{\beta}}{\sum_{j=1}^{n}[\tau_{2j,2j+1}(t)]^{\alpha}[\eta_{2j,2j+1}]^{\beta}}, 1\right). \tag{4.1}$$

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

$$\sum_{i=1}^{n} p_{(i,i+1)}(t) = \sum_{i=1}^{n}\left[\min\left(\frac{\theta n[\tau_{2i,2i+1}(t)]^{\alpha}[\eta_{2i,2i+1}]^{\beta}}{\sum_{j=1}^{n}[\tau_{2j,2j+1}(t)]^{\alpha}[\eta_{2j,2j+1}]^{\beta}}, 1\right)\right]. \tag{4.2}$$

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

$$\sum_{i=1}^{n} p_{(i,i+1)}(t) \leq \sum_{i=1}^{n}\left(\frac{\theta n[\tau_{2i,2i+1}(t)]^{\alpha}[\eta_{2i,2i+1}]^{\beta}}{\sum_{j=1}^{n}[\tau_{2j,2j+1}(t)]^{\alpha}[\eta_{2j,2j+1}]^{\beta}}\right) = \theta n. \tag{4.3}$$

Therefore, the expected number of STS groups to be integrated is upper bounded

by θn. With the increase of θ, 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 θ can also affect the computational time of the proposed optimizer. A high value of θ 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 $\theta \in [0.2, 0.4]$.

4.3 Experiments

4.3.1 Experimental Setting

Table 4.1: Details of the models utilized in the experiments.

Problems	Number of layers	Total number of segments	Average number of segments on a layer	Maximum number of segments in layers	Minimum number of segments in layers	SD of the number of segments in layers
3DHackerTest	271	150670	555.99	7690	3	720.68
Debailey_x10	1035	424420	410.06	1142	3	266.09
TortureTestV2	523	361580	691.36	1715	3	600.59
UltimakerRobot _support_2015	877	481190	548.68	1939	3	200.02
ctrlV_3D_test	236	184380	781.25	2538	3	460.4
dragon_65_tilted_large	692	839210	1212.7	2769	3	544.81
hold_test	98	190570	1944.6	2484	3	260.49
testModel	598	110560	184.87	406	3	125.93

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 [61] 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 [64,66] 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. Details of the models utilized in the experiments is shown in Table 4.1. Different path optimization modules were utilized for comparison purposes. Cura is integrated with a greedy-based optimizer [61], 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 [67]. 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 [41]. 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 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 [31], 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 [31] outperforms those utilized other reasonable parameter sets. A summary of the parameters used in this work is shown in Table 4.2.

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.

4.3.2 Experimental Results

Accuracies

Experiments were conducted to evaluate the dimensional accuracy of printed models using print plans optimized by the modified ACO.

Similar to [69], in this set of experiments, the model "testcube_25mm" [68] 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. 4.4.

Table 4.2: Parameters utilized in generic ACO, the proposed modified ACO-based optimizers, and Cura.

Number of iterations	8
Number of ants	8
α	1
β	5
ρ	0.5
θ	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

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. 4.4, which were obtained using a "Mitutoyo Digital Caliper 500-196-20".

According to Fig. 4.4, 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.

Post-processing Time

All optimizers under test were executed 50 times to obtain their average run times. Results are given in TABLE 4.3. According to the results, when the parameters of the modified ACO optimizer was configured as shown in TABLE 4.2, it yields comparable post-processing times as Christofides' and Frederickson's algorithms. Furthermore, when comparing with the generic ACO optimizer under

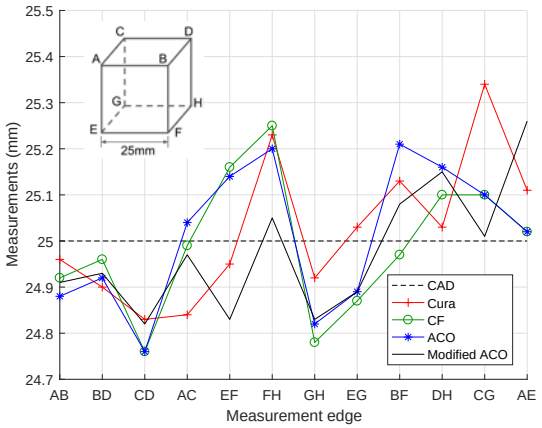


Figure 4.4: An illustration (top left) of the labelled cube [68] used in the experiment and the edge measurements of the printed models using print plans from different optimizers. The dotted line represents the edges' length of the CAD model.

Table 4.3: 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

the same configurations, the proposed one can reduce the overall post-processing time significantly by 34.93%.

Estimated Print Time

Results obtained using the GCodeAnalysor are presented in TABLE 4.4. 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 2.2.1, Christofides' and Frederickson's algorithms

are deterministic algorithms. Therefore, the optimizer with CF always generates identical results. According to TABLE 4.4, 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 were 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 μ'_{ACO} and σ'_{ACO}, respectively. Similarly, the mean and SD of the estimated print time using the generic ACO are represented respectively by μ_{ACO} and σ_{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.

Table 4.4: 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

Table 4.5: 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

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 [63]. 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 4.5. 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 4.5 concur with those observations 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.

Visual Quality

The visual quality was verified according to the amount of strings, which are residual materials left on the surface of a printed model. CAD of the model "hold_test" [66] is shown in Fig.4.5. Pictures of the model "hold_test" [66] fabricated using print plans from different optimizers are shown in Figs. 4.6. In this experiment, to evaluate the strings problem, the residual materials formed on the surface of the print models were measured. Only residual materials with any of its dimensions longer than 0.5mm are regarded as strings and they are highlighted

with red boxes in Fig 4.6. There are multiple factors that would affect the number of strings formed on the surface of the models, including the route of the printing nozzle, the specification of the hardware, and the configurations of the 3D printer. Strings problem could be less severe for more advanced 3D printers with better control mechanisms on the flow of filament. The contribution of this work, however, will greatly enhance the printing quality of consumer/pro-sumer-graded printers without any hardware modifications.

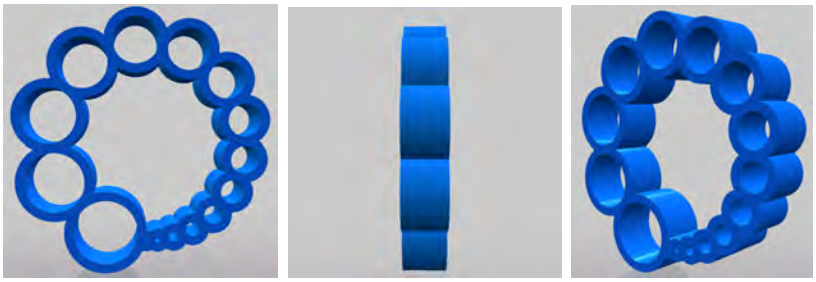
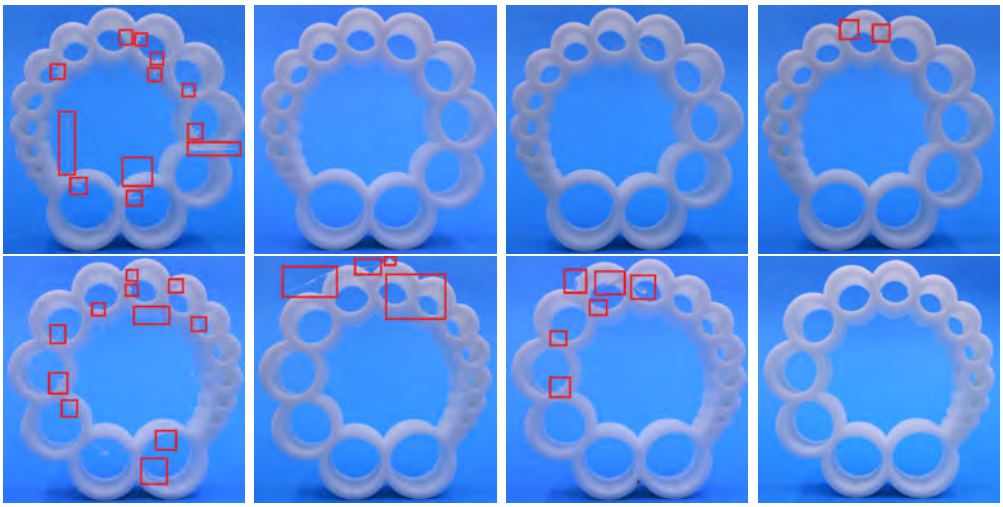


Figure 4.5: CAD of the model "hold_test". [66]



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

Figure 4.6: 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.

According to Fig. 4.6, 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 the 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.

4.4 Discussion

According to the post-processing times given in TABLE 4.3, 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 4.2.4, an immature convergence may lead to incorrect STS grouping and degrade the solution quality. Based on the estimated print time given in TABLE 4.4, 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 an 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 4.3, 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 Frederickson's algorithms cannot be benefited from parallel processing as these two algorithms consist of sequential procedures as mentioned in Section 2.2.1.

4.5 Conclusion

In this chapter, an ACO based tool-path optimizer is proposed for finding desirable tool-paths in 3D printing applications, which can both shorten the actual print time and improve the visual quality of printed objects. The proposed optimizer has a two-stage 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 experimental results have shown that the proposed optimizer together with the proposed ACO solver can yield shorter model build time than other optimizers under test. Furthermore, experimental 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.

Chapter 5

Efficient Tool-path Optimizer

In the previous chapter, a tool-path optimizer based on a modified ACO is proposed, which exploits unique properties in 3D printing to accelerate the 3D printing processes and the post-processing duration. In this chapter, a computationally efficient heuristic search algorithm is proposed to find fast routes and mitigate overheads in printing processes.

The optimizer being introduced in this chapter utilizes a detour search algorithm and a local search algorithm to significantly speed up printing processes and reduce the post-processing time simultaneously. The proposed tool-path optimizer was evaluated using both simulations and actual 3D printing experiments. Results are presented in Section 5.3. The results are analysed and discussed in Section 5.4. Concluding summary are given in Section 5.5.

5.1 Introduction

The k–opt local search algorithm was first proposed to refine sub-optimal TSP solutions [51]. It was later modified by Hertz [52] to work with other URPP solvers. Recently, a k–opt algorithm was utilized to solve a nozzle path planning problem [30]. Their study showed that the k–opt algorithm can shorten fabrication

time. The authors found that the fabrication time can be further reduced by using a 3-opt instead of a 2-opt algorithm. However, the processing time required by using a 3-opt algorithm is significantly longer than its counterpart. Their study suggested that it is extremely time-consuming or even impractical to utilize a conventional 3-opt algorithm on solving real-life nozzle tool-path planning problems. Details of a conventional k–opt algorithm are introduced as follows.

In a k–opt algorithm, a given tour is broken down into k fragments and reconnected using different feasible combinations. The tour is updated whenever an improvement is found within a search process. Here, an improvement refers to a modification that yields a cost reduction of a tour. The process repeats until no further improvements can be found.

Similar to [53], when a k–opt algorithm is used to solve a URPP with n required edges, the total number of combinations to be evaluated in the search space in each iteration is

$$\frac{n!\ 2^{k-1}}{k!\ (n-k)!}. \tag{5.1}$$

The total number of combinations increases exponentially with k and n. Previous studies [53, 55] showed that 2–opt and 3–opt algorithms can deliver promising results. However, 3–opt algorithms usually requires significantly longer processing times, which may not be applicable in practical production scenarios. For example, for a URPP with 100 required edges, the number of combinations increases from 9900 to 646800 when k increases from 2 to 3.

5.2 The Proposed Method

To accelerate printing processes, two algorithms are proposed in this section which can reduce the number of transitions with retraction (TwR) and shorten the total transition duration. Their operating procedures are introduced in this section.

5.2.1 Detour Search Algorithm

A heuristic searching algorithm is proposed to use detours to replace unnecessary TwR, which can shorten the overall fabrication duration. Here, a *tour* represents a valid solution that traversed all required segments. A *detour* is a directed route comprises a series of print segments connected with transitions, which traverses only some of the required segments. A detour starts and ends with the two extremities of the TwR to be replaced. An interim detour, which is formed on the fly, represents a route that starts with an extremity of a TwR but has not yet reached the other extremity of the same TwR. Each interim detour is associated with a *preserved route*, which is a directed route. A preserved route includes all print segments which are currently not included in its corresponding interim detour. During the detour search process, the interim detour will expand by connecting new print segments to it. At the same time, the corresponding print segments in its preserved route are removed, transitions are then added accordingly to ensure all print segments are visited by the preserved route or the interim detour exactly once. Therefore, a feasible solution can always be constructed by combining an interim detour and its corresponding preserved route.

The proposed detour search algorithm creates and expands interim detours iteratively and eventually find a detour with the minimum time cost to replace the TwR provided that such a detour exists. The procedures of the detour search algorithm are explained with the aid of the following example. To ease explaining, a route is expressed using a set of print segments and transitions. Therefore, the tour in Fig. 5.1(a) is expressed as $\langle \mathrm{RA}, \mathrm{AB}, \mathrm{BC}, \cdots, \mathrm{IQ}, \mathrm{QR} \rangle$.

1. The algorithm starts with an input tour. The algorithm searches all TwR in the solution. For each TwR, the input tour is broken down into a directed route by removing the selected TwR. Therefore, the two extremities of the TwR are the start point and end points of the directed route. As shown

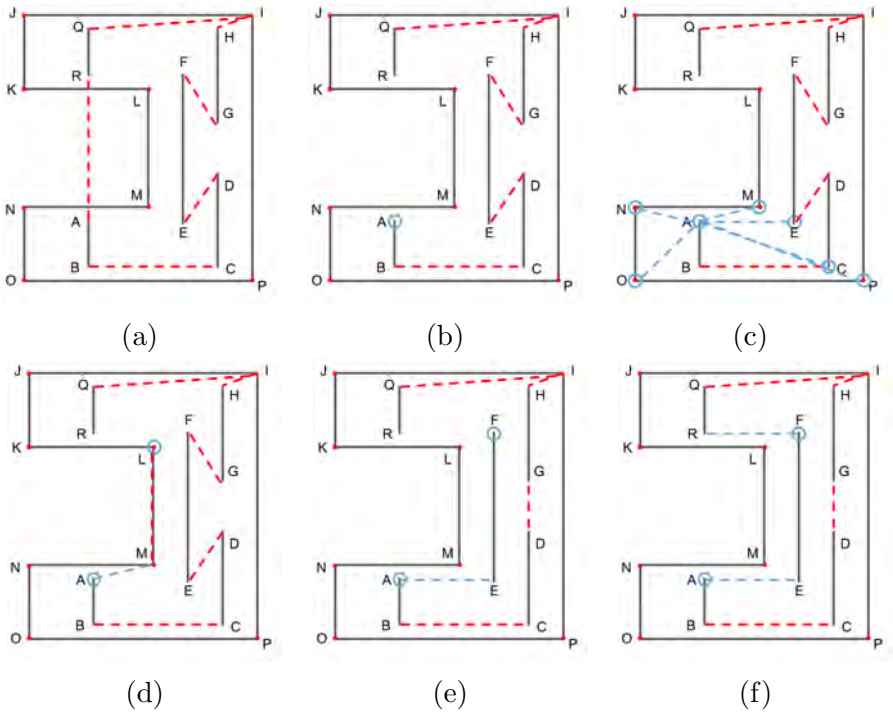


Figure 5.1: Illustrations showing (a) a print plan, (b) a directed route with a selected transition with retraction removed, (c) all available interim detours that starts at A, (d) an interim detour begins with traversing through AM, (e) an interim detour begins with traversing through AE, and (f) a resultant tour. Black, red dotted, and blue dotted lines represent print segments, transitions, and interim detour, respectively. Blue circles denote the start or end point of an interim route. The extremities of print segments are labelled.

in Fig. 5.1(a), AR is the only TwR on the input tour, so it is selected and removed from the tour. The points A and R will be used as the start and end points of the detour generated in the next step, respectively. It is worth to note that a TwR associated with the highest cost is selected first if there are more than one TwR on the input tour.

2. An initial interim detour is constructed at the beginning of the search, *i.e.* at point A in Fig. 5.1(a). The interim detour $\langle \mathrm{AA} \rangle$ is a detour with no displacement as it starts and ends at the same point A. Its corresponding preserved route form a pair of routes. This pair of routes is then added to an OPEN list. In this work, all generated pair of routes are stored in an OPEN list. Each pair of routes contains an interim detour and its corresponding

preserved route. Furthermore, the OPEN list contains only the pairs that can be further expanded to form a valid detour but have not yet expanded.

The preserved route of the interim detour $\langle \mathrm{AA} \rangle$ is a directed route obtained from the input tour but with the TwR AR removed, *i.e.*, $\langle \mathrm{AB}, \mathrm{CD}, \cdots, \mathrm{GH}, \mathrm{QR} \rangle$.

3. At each iteration of the searching process, an interim detour with the minimum time cost in the OPEN list is selected. The time cost of an interim detour represents the time cost required by the nozzle to traverse from the start point to the last point of that interim detour. The algorithm then expands the selected interim detour as follows.

   Here, the interim detour $\langle \mathrm{AA} \rangle$ is selected from the OPEN list. The algorithm expands it by generating new interim detours. The algorithm finds one print segment in the corresponding preserved route and appends it to the selected interim detour with a proper transition as a connection. In this process, a print segment will not be selected if a TwR is required between that print segment and the last point of the interim detour. All possible options are shown in Fig. 5.1(c). It is worth to note that, the pair of the interim detour $\langle \mathrm{AA} \rangle$ and its corresponding preserved route are removed from the OPEN list as they have already been expanded.

4. The algorithm generates a new interim detour based on each possible option. Suppose the print segment ML is selected, a new interim detour is generated by appending ML to it, *i.e.* $\langle \mathrm{AA}, \mathrm{AM}, \mathrm{ML} \rangle$. The print segment ML is then removed from the preserved route. Afterward, a new transition is introduced to keep the preserved route connected. The new interim detour and its corresponding preserved route as shown in Fig. 5.1(d) are added to the OPEN list.

Alternatively, if EF is selected, a new interim detour is generated, *i.e.* $\langle \mathrm{AA}, \mathrm{AE}, \mathrm{EF} \rangle$. Similarly, a new transition is added to the preserved route to connect the print segments CD and GH. Therefore, the corresponding preserved route becomes $\langle \mathrm{AB}, \mathrm{BC}, \mathrm{CD}, \mathrm{DG}, \mathrm{GH}, \cdots, \mathrm{IQ}, \mathrm{QR} \rangle$ as shown in Fig. 5.1(e). The results are added to the OPEN list. The algorithm repeats the processes for all the possible options as shown in Fig. 5.1(c).

5. In the next step, the algorithm finds an interim detour with the minimum cost in the OPEN list and connects that to the end point. Different from the conventional A* algorithm, the proposed algorithm selects a print segment $e = (i, j)$ that minimizes

$$f(e) = g'(e) + h(j), \tag{5.2}$$

$$g'(e) = g(i) + \Delta(e), \tag{5.3}$$

where $g(i)$ denotes the time cost for traversing from the start point to the point i, and $h(j)$ is the estimated time cost for traversing from point j to the end point. Here, $g(i)$ calculates the exact value required by the nozzle to traverse an interim detour using (1.6). The value of $h(j)$ is calculated using (1.8). By using the triangle inequality and $t_T(e) \geq t_M(e)$, it is known that $h(j)$ is upper bounded by the actual time required for the nozzle to traverse from point j to the end point. Therefore, $h(j)$ is admissible.

In this work, $\Delta(e)$ represents the difference in time cost of the preserved route when e is removed from it. Therefore, for the interim detour in Fig. 5.1(d), $\Delta(\mathrm{AM}) = t_T(\mathrm{LM})$. Similarly, for the interim detour in Fig. 5.1(e), $\Delta(\mathrm{AE}) = t_T(\mathrm{DG}) - t_T(\mathrm{DE}) - t_T(\mathrm{FG})$. It is given that $\Delta(\mathrm{AE}) \leq \Delta(\mathrm{AM})$, such that the interim detour in Fig. 5.1(e) is selected. The algorithm then checks if the selected interim detour reaches the end point without creating

new TwR. As shown in Fig. 5.1(f), such a route exists. Therefore, a resultant tour is generated by combining the interim detour and its corresponding preserved route. Otherwise, the algorithm begins its next iteration and goes back to step 2. If the OPEN list is empty, there does not exist any interim detour that can be further expanded to find a detour. The algorithm keeps the TwR in the solution and proceeds to another TwR until no further changes can be made.

5.2.2 An Efficient Implementation of the k–opt Local Search

In this work, an efficient implementation of the k–opt local search algorithm is developed based on the characteristics in NPPP in order to accelerate the optimization process and generate solutions with the same quality. The proposed method can significantly shorten the optimization process and generate solutions with the same qualities to those obtained using the conventional k–opt implementation. In this work, the transitions $E \setminus E_r$ in the NPPP are classified exclusively into one of the subsets of E as follows.

$$E_B = \big\{e\colon e \in E \setminus E_r,\ t_R(e) = 0,\ d(e) > 0\big\},$$

$$E_C = \big\{e\colon e \in E \setminus E_r,\ t_R(e) = 0,\ d(e) = 0\big\}.$$

To ensure results of the proposed planner will always be better or the same as an ordinary k-opt local search algorithm, two theorems have been developed which exploit the unique properties in 3D printing applications. Their proofs are provided in the APPENDIX.

Theorem 1. The combinations which are composed of one transition in E_B and $(k-1)$ transitions in E_C do not deliver improvement to the solution.

Algorithm 1: Detour search algorithm

Data: A valid solution T as input.
Result: A solution in which unncessary transitions with retraction are removed.

isUpdated $\leftarrow$ TRUE;
while isUpdated **do**
  isUpdated $\leftarrow$ FALSE;
  $S_r \leftarrow e \in T \setminus E_r, t_R(e) \neq 0$;
  if $S_r \neq \emptyset$ **then**
    for each transition with retraction $(A, B) \in S_r$ **do**
      OPEN $\leftarrow \emptyset$;
      Generate initial interim detour $i_{\text{init}} \leftarrow \langle A \rangle$;
      Genrates preserved route p_{init} based on i_{init} and T;
      Append $(i_{\text{init}}, p_{\text{init}})$ to OPEN;
      isSearching $\leftarrow$ TRUE;
      while isSearching or OPEN $\neq \emptyset$ **do**
        Find a pair (i_a, p_a) in OPEN where i_a have the lowest cost in OPEN;
        Remove (i_a, p_a) from OPEN;
        for each $e : e \in p_a, e \in E_r$ **do**
          if e_t exists where $e_t \in E \setminus E_r, t_R(e_t) = 0$, and e_t connects i_a to e **then**
            Create new interim detour i_{new} based on i_a and e_t;
            Create new preserved route p_{new} based on i_{new} and e_t;
            Append $(i_{\text{new}}, p_{\text{new}})$ to OPEN;
          end
        end
        for each interim detour i in OPEN **do**
          if i connects to B using a transition t without retraction **then**
            isSearching $\leftarrow$ FALSE;
            isUpdated $\leftarrow$ TRUE;
            Create a detour d using i and t;
            Update T using d;
          end
        end
      end
    end
  end
end
return T

Theorem 2. Combinations which are only composed of transitions in E_C do not deliver improvement to the solution.

The new implementation is developed based on these two theorems. According to Theorems 1 and 2, the two kinds of combinations specified are excluded from evaluation processes as those combinations can never deliver improvement to a solution. Therefore, the total number of combinations to be evaluated in the search space in each iteration becomes

$$\frac{n!\ 2^{k-1}}{k!\ (n-k)!} - \frac{|E_B|(|E_C|)!\ 2^{k-1}}{(k-1)!(|E_C|-k+1)!} - \frac{|E_C|!\ 2^{k-1}}{k!\ (|E_C|-k)!}.$$

The number of combinations to be evaluated is upper bounded by the number of combinations generated by the conventional k–opt implementation which is given in (5.1). The saving is significant if the value of $|E_B|$ or $|E_C|$ is large. Here, $|E_B|$ and $|E_C|$ represent the number of transitions in the subset E_B and E_C, respectively.

The proposed method utilizes both the efficient implementation of k–opt local search and the detour search algorithms to accelerate the 3D printing process. It proceeds in an iterative manner. An iteration is started by executing the detour search algorithm on an input solution. The solution is modified such that TwR in the solution are replaced by available detours. The detour search algorithm proceeds until no detour can further be found. The solution is then further optimized using the efficient k–opt local search algorithm. If an improvement is found by the search algorithm, the iteration proceeds. Otherwise, the process terminates and returns the solution.

5.3 Experiments

5.3.1 Experimental Setting

Experiments were conducted to evaluate the performance of the proposed algorithm. A total of ten 3D models given in [70] were selected in the experiments. Details on the selected models are shown in TABLE 5.1. Some of the models are rotated or scaled to ease the evaluations. The 3D models were first sliced using Cura-15.04.6 [61] with its default settings. Furthermore, retractions were enabled; the vertical hop and retraction lengths are 0.1 mm and 6.5 mm, respectively. The filling density for infilling processes was set to 10%.

Different path optimization modules were utilized for comparison purposes. In the experiments, Frederickson's algorithm (Fred) was utilized to generate initial solutions for some optimization modules under test. Ten different combinations of optimization algorithms were utilized, including Frederickson's algorithm, 2–opt algorithms, 3–opt algorithms, ACO, GA, and the proposed method.

In the experiment, the proposed method starts with using Frederickson's algorithm and followed by the algorithms introduced in Sections 5.2.1 and 5.2.2. In this work, each meta-heuristic (*i.e.* ACO and GA) was executed with three different tuning parameters to give a better comparison. For ACO, the values of α, β, and ρ are 1, 3, and 0.2, respectively [50]. The numbers of both ants and iterations for ACO 1, 2, and 3 are 100, 200, and 300, respectively. For GA, the probabilities of mutation and crossover are 0.15 and 0.8, respectively [71]. The numbers of both individuals and iterations for GA 1, 2, and 3 are 100, 200, and 300, respectively.

In the experiments, the estimated print time and the number of TwR of print plans were obtained using an open-source software GCode-Analysor-1.0 [67]. Moreover, all programs were executed on a computer with Intel Core i7 processors,

Table 5.1: Details of the selected 3D models [70].

Names	Number of layers	Number of print segments
Articulated butterfly	78	180496
Clamps	128	216171
Customizable linear bushing	348	99351
Indispensable dispenser	249	147665
Labyrinth gift box	498	103721
Lowest poly thinker	598	94779
Russian doll maze puzzle box	583	293644
USB stick holder	1231	507314
Velociraptor business card	8	39290
XYZ 20 mm calibration cube	198	17469

16 GB RAM, and Ubuntu 16.04.

Experiments were conducted to assess the performance of the proposed method in three different aspects: the estimate print time, the post-processing time, and the number of TwR. Experimental results are given in TABLES 5.2, 5.3, and 5.4. Here, the post-processing time refers to the time required by a method to generate an optimized print plan.

Each set of experiments was executed for 30 times in order to obtain the average values of the estimate print times and the post-processing times. In order to verify the actual performance of the proposed method, ten 3D models were further printed using a WiseMaker 3D Printer W250 [63]. The print times were recorded. Experimental results are shown in TABLE 5.5.

5.3.2 Experimental Results

According to TABLE 5.2, the proposed method can on average reduce the estimated print time by 26.10% when comparing to that of Cura. Its saving in estimated print time is the highest among all tested methods. The proposed method achieved a maximum saving of 36.13% estimated print time for the model "Indispensable dispenser". The proposed method can deliver the same or much higher

Table 5.2: The estimated print time (s) of print plans obtained using different optimization modules.

Models	Cura	Fred	Fred 2–opt	Fred 3–opt	ACO 1	ACO 2	ACO 3	GA 1	GA 2	GA 3	Proposed
Articulated butterfly	18046	14361	12764	12585	12773	12743	12734	13530	13441	13385	**12572**
Clamps	8762	7983	6761	6582	7055	6986	6967	7585	7533	7532	**6538**
Customizable linear bushing	2098	2098	1619	1530	2098	2098	2098	2098	2098	2098	**1520**
Indispensable dispenser	4122	3788	2967	2727	3616	3528	3519	3998	3980	3979	**2632**
Labyrinth gift box	2962	2590	2269	2208	2962	2962	2962	2962	2962	2962	**2198**
Lowest poly thinker	6408	5857	5501	**5475**	5502	5497	5495	5681	5658	5644	**5475**
Russian doll maze puzzle box	8806	8806	6759	6171	8806	7990	7952	8806	8806	8806	**6059**
USB stick holder	23122	22225	20063	19609	20945	20721	20645	22358	22319	22284	**19558**
Velociraptor business card	2421	2421	1965	1844	2111	2081	2079	2421	2421	2421	**1828**
XYZ 20 mm calibration cube	2732	2005	1915	**1910**	1911	1911	**1910**	1936	1935	1936	**1910**

Table 5.3: The post-processing time (s) required using different optimization modules.

Models	Fred	Fred 2–opt	Fred 3–opt	ACO 1	ACO 2	ACO 3	GA 1	GA 2	GA 3	Proposed
Articulated butterfly	**11.31**	12.05	56.20	188.79	719.96	1653.43	106.20	390.35	885.51	28.84
Clamps	**77.32**	87.59	1954.34	327.38	1064.04	2357.86	322.66	1105.13	2459.01	304.73
Customizable linear bushing	**12.75**	13.20	56.21	136.40	489.63	1115.99	132.15	511.90	1145.16	16.09
Indispensable dispenser	**116.50**	128.28	3831.64	291.14	836.71	1805.11	370.78	1277.68	2844.22	308.38
Labyrinth gift box	**7.03**	7.29	35.32	128.85	474.65	1091.38	103.78	402.83	905.26	9.96
Lowest poly thinker	**2.31**	2.56	12.41	116.67	450.84	1032.92	61.08	233.80	533.73	4.18
Russian doll maze puzzle box	**104.65**	114.29	1517.60	467.89	1513.47	3349.04	574.82	2108.79	4684.45	220.24
USB stick holder	**159.79**	177.38	2642.64	821.42	2766.04	6230.21	907.99	3377.88	7474.99	381.00
Velociraptor business card	**10.48**	11.84	247.86	57.40	201.51	463.39	46.73	157.23	348.88	57.38
XYZ 20 mm calibration cube	**0.23**	0.25	1.17	15.66	62.92	144.05	6.80	27.29	62.74	1.08

performance than other methods in all the models under test. Note that GA and ACO failed to further reduce the estimated print time of the initial solution on some models.

According to TABLE 5.3, it can be observed that Fred 2–opt needed on average 8.9% longer post-processing time when comparing to that with Frederickson's algorithm solely and Fred 3–opt needed on average 1,277% longer post-processing time. On the other hand, the proposed method, which starts with Frederickson's algorithm, followed by the proposed efficient 3–opt algorithm and effective lo-

Table 5.4: The number of transitions with retraction using different optimization modules.

Models	Cura	Fred	Fred 2–opt	Fred 3–opt	ACO 1	ACO 2	ACO 3	GA 1	GA 2	GA 3	Proposed
Articulated butterfly	12121	5069	2124	1844	2151	2112	2100	3418	3233	3119	**1824**
Clamps	5310	3967	1699	1410	2231	2113	2077	3232	3130	3130	**1318**
Customizable linear bushing	1383	1383	543	372	1383	1383	1383	1383	1383	1383	**352**
Indispensable dispenser	3490	2994	1440	988	2609	2446	2429	3339	3303	3302	**803**
Labyrinth gift box	1964	1409	799	682	1964	1964	1964	1964	1964	1964	**661**
Lowest poly thinker	3268	2391	1742	1714	1743	1741	1739	1981	1932	1910	**1713**
Russian doll maze puzzle box	6079	6079	2396	1274	4941	4579	4504	6079	6079	6079	**1056**
USB stick holder	6306	6116	2096	1347	3327	2969	2845	5822	5742	5675	**1251**
Velociraptor business card	1353	1353	515	302	761	707	700	1353	1353	1353	**270**
XYZ 20 mm calibration cube	1734	374	205	202	**201**	**201**	**201**	226	225	224	202

Table 5.5: The actual print time (s) using the print plans obtained using Cura and the proposed method.

Models	Cura	Proposed
Articulated butterfly	17674	12990
Clamps	8787	6609
Customizable linear bushing	2604	2080
Indispensable dispenser	4229	2909
Labyrinth gift box	3718	3041
Lowest poly thinker	6566	5656
Russian doll maze puzzle box	8744	6196
USB stick holder	23016	19164
Velociraptor business card	3081	2464
XYZ 20 mm calibration cube	2610	1899

cal search algorithm, executed significantly faster. The proposed method only required on average 30.96% of the post-processing time of Fred 3–opt.

Furthermore, when considering the methods using ACO or GA, it can be observed that the post-processing time increased rapidly with the increase in the numbers of iterations and search agents. For instance, ACO 3 and GA 3 respectively required on average 67 and 44 times longer post-processing time than the proposed method.

Fig. 5.2 shows the post-processing time required by using different tool-path optimizer with respect to the total number of segments given in TABLE 5.1.

It can be observed that, in general, the post-processing time is highly related to the total number of segments in a model. However, there are some exceptions. For example, the model “Indispensable dispenser” required on average longer post-processing times than the model “Articulated buttery”, which are composed of 147665 and 180496, respectively. It indicated that some factors exist which would alter the post-processing time. Investigating the relation between post-processing and the model is considered as out of scope in this study. In general, with an increase in the total number of segments, the post-processing time required by Frederickson’s algorithm solely and Fred 2-opt increased relatively slower than

other methods. The proposed method delivered similar performance.

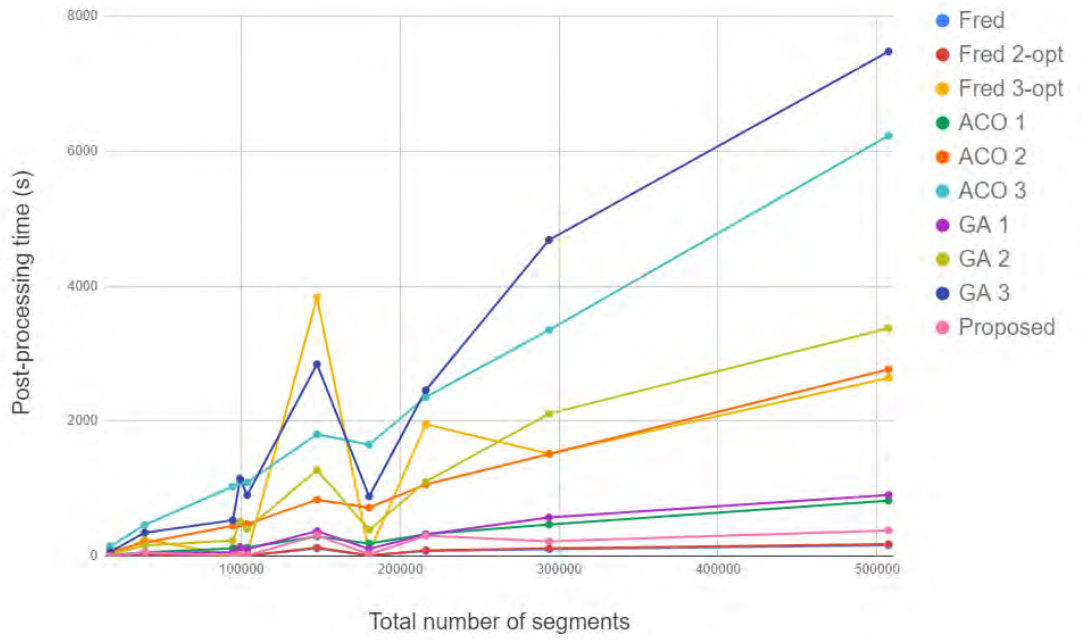


Figure 5.2: The post-processing time (s) required using different optimization modules.

To evaluate the methods under a practical scenario, the sums of the average estimated print time and the average post-processing time for each model were used for comparison. According to TABLES 5.2 and 5.3, the proposed method delivered the best performance on total time saving among all tested methods on all models under test. The proposed method can on average yield around 24.21% total time saving compared to that of Cura. The methods ACO 3, GA 1, GA 2, and GA 3 could not achieve time saving when the post-processing time is considered. For these methods, the time spent on post-processing is significantly longer than the saving in the printing processes. These methods may not be practical in actual scenarios where post-processing time should also be considered.

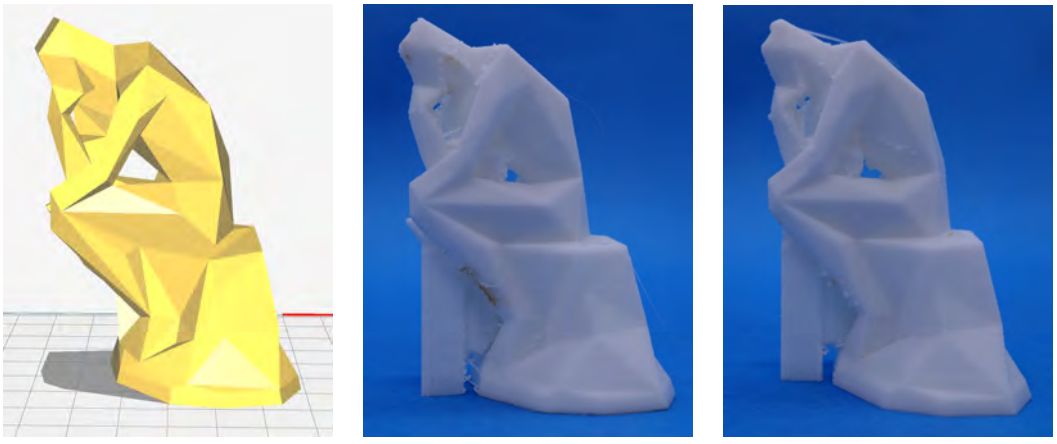
According to TABLE 5.4, it can be observed that the proposed method delivered the best results amount all tested methods. It has reduced around 75.68% on the number of TwR for all models tested when comparing to that of Cura. The meta-heuristics methods cannot efficiently reduce the number of transitions

with retraction in the experiments when comparing to the proposed method. Even though ACO 3 has delivered the best overall result among all tested meta-heuristics, it can only reduce the number of TwR by around 43.82% when comparing to that of Cura.

According to the experimental results reported in TABLE 5.5, the proposed method can yield a total time saving of around 22.78% when comparing with solutions from Cura. The results concur with those shown in TABLES 5.2 and 5.3.

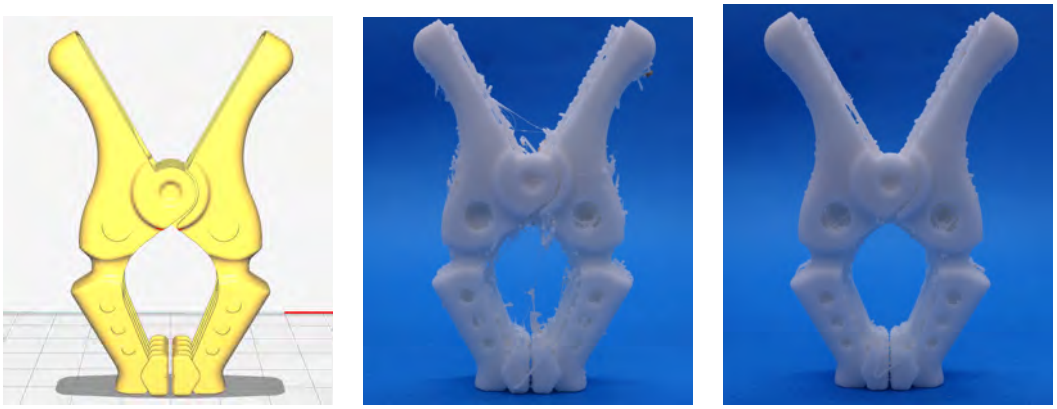
Visual Quality

Strings formed due to unwanted filament dripping on the surface of the model degrade the visual quality of the printed parts. Although the number of strings on models printed cannot be quantified directly, their visual quality can be evaluated by comparing them qualitatively. Figs. 5.3, 5.4, 5.5, and 5.6 show the CADs and the photographs of the printed parts of the models "Lowest poly thinker", "Clamps", "Indispensable dispenser", and "USB stick holder".



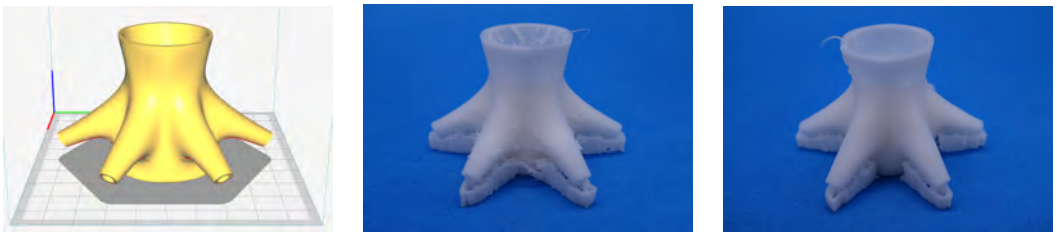
(a) (b) (c)

Figure 5.3: (a) CAD of the model "Lowest poly thinker". The model printed together with the support structure using (b) Cura and (c) the proposed method.



(a) (b) (c)

Figure 5.4: (a) CAD of the model "Clamps" [70]. The model printed together with the support structure using (b) Cura and (c) the proposed method.

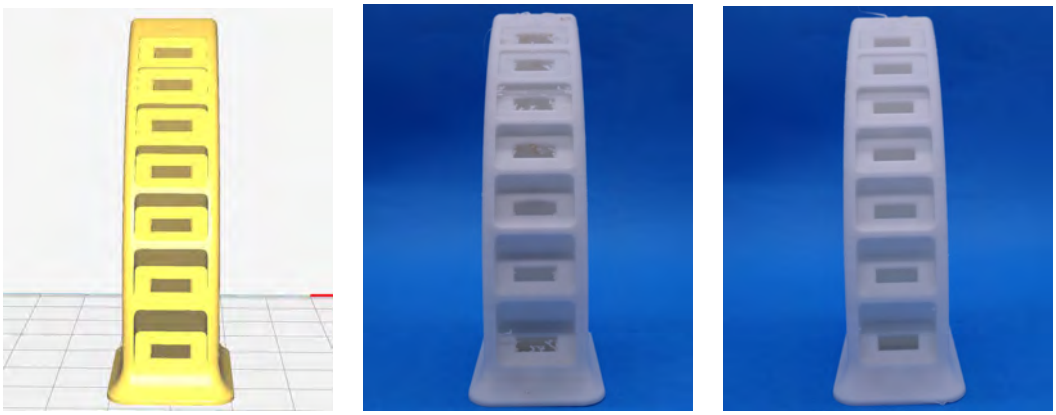


(a) (b) (c)

Figure 5.5: (a) CAD of the model "Indispensable Dispenser" [70]. The model printed together with the support structure using (b) Cura and (c) the proposed method.

5.4 Discussion

When considering the print plans generated using Fred 3–opt, it can be observed that their estimated print times were close to that of the proposed method. However, according to TABLE 5.3, Fred 3–opt needed on average 5 times of post-processing time to that of the proposed method in the experiment. In general, the post-processing time required by Fred 3–opt increases with the number of print segments in the models. It can also be observed that the post-processing time required by Fred 3–opt increased rapidly with the number of print segments. In contrast, the post-processing time of the proposed method increases more gently when comparing to that of Fred 3–opt.



(a) (b) (c)

Figure 5.6: (a) CAD of the model "USB stick holder" [70]. The model printed together with the support structure using (b) Cura and (c) the proposed method.

According to TABLE 5.4, the proposed method delivered the best performance by reducing on average 75.68% of the number of TwR when comparing to that of Cura. It can also be observed that Fred 2–opt and Fred 3–opt outperform the meta-heuristics methods.

For meta-heuristic methods, in this work, a maximum of 300 iterations were utilized. In general, the quality of solutions generated by typical meta-heuristics improves with the growth of iterations. As expected, the estimated print time, obtained by using larger numbers of iterations can be further reduced in most cases. However, such an advantage has been diminished quickly as the processing time of meta-heuristic methods increases with its agent and iteration numbers.

According to Table. 5.2, there is an increase in post-processing time for the model "Indispensable dispenser", which has a total of 147665 print segments. In general, the post-processing time required by an optimizer to process a model increases linearly with the number of print segments. However, for the model "Indispensable dispenser", it can be observed that there are no disjoint parts on most of the layers of this model. There was not enough rooms for the segment-consolidation scheme proposed in Section 3.2.1 to perform well.

In general, the proposed method can deliver promising results in accelerating

the printing process within a reasonable post-processing time. Moreover, to obtain further improvement, the proposed method can work with meta-heuristics, such as the ACO-based optimizer proposed in Chapter 4. The proposed method is designed to improve an initial solution, it can work with typical meta-heuristics to improve their performances. For example, when implementing with GA, the proposed method can be executed on the final solution or executed at the end of every iteration. However, the proposed method should only be applied to some of the iterations or intermediate solutions to prevent premature convergence in meta-heuristics *i.e.* 10% of the population. Similar procedures can be applied to improve the performance of ACO and other meta-heuristics which operate in an iterative manner.

5.5 Conclusion

In this chapter, an efficient nozzle path planner is proposed for accelerating 3D printing processes. For typical 3D printers, a series of time-costly operations is required to avoid unwanted molten filament from dripping out of the nozzle and forming residues on the surface of the model. A new algorithm is proposed to exploit replacement paths that can eliminate some of those operations. Furthermore, a customized local search algorithm is designed and implemented to search for those replacement paths which needs a significantly shorter processing time than conventional algorithms. Experimental results show that model print times can be significantly reduced by applying the proposed method as printed objects can be fabricated using a faster plan. The proposed method can achieve a maximum saving in print time of around 30% when compared to its counterparts. With proper formulations, the proposed planner can be applied in other computer numerical control-based applications.

Chapter 6

Conclusions and Suggestions for Future Work

In this chapter, the main contributions achieved in this thesis are summarized. Some suggestions for future research are also discussed.

6.1 Main Contributions of the Thesis

In previous decades, many researches have been done on improving the performances of additive manufacturing in different aspects, including printing accuracy, surface finishing, physical strength, tool-path generation, tool-path planning, and etc. Although much work studied the tool-path planning problem in industrial applications, such as aerosol printing processes and spray forming processes, an in-depth research focusing on the tool-path planning problem in 3D printing, which is also known as layered additive manufacturing, has not been fully established.

In this thesis, tool-path optimizers are proposed to solve the tool-path planning problem in 3D printing processes. The main objective is to minimize the fabrication time by planning the printing sequence and other machining opera-

tions. To print parts with high structural integrity, the print segments, where the printing nozzle deposits material, should not be reoriented or skipped. However, the printing sequence can be altered to affect the total time for the printing nozzle to visit all the print segments via transitions. Therefore, fabrication time can be shortened by minimizing the time spent on transitions without affecting the accuracy and the physical properties of the printed model. Moreover, minimizing the duration of the tool-path planning process is also critical. An optimizer is impractical if the post-processing time required by the optimizer is longer than the time-saving in print time obtained by the optimizer.

The main contributions of this thesis are summarized as follows.

1. A fundamental formulation of the tool-path planning problem in 3D printing applications and essential modifications are introduced to enable TSP or URPP solvers to be utilized in solving the problem. Also, a segment-consolidation scheme is developed to speed up the optimization process, the tool-path planning is carried out at the inter-partitions level and the intra-partition level sequentially. The optimizer finds a fast sequence for visiting all partitions, which are referring to the sets of print segments within different boundaries on the same layer. Also, massive short and chained print segments, which are utilized to represent curves on a print layer, are consolidated into replacement segments during the optimization process. Replacement segments are reverted to the corresponding chains of print segments at the end of the optimization process. Therefore, by adopting the above procedures, the computational complexity of the tool-path planning process can be reduced without introducing significant impacts on the solution quality.

2. An ACO-based tool-path optimizer is proposed to further accelerate the printing processes. ACO is a nature-inspired meta-heuristic and has been

utilized to solve TSP and URPP. Unlike deterministic algorithms, ACO has a stochastic mechanism, such that, ACO can generate, in theory, better solutions monotonically with the growth of iterations. Essential modifications are introduced to enable the tool-path planning in 3D printing application can be solved by ACO based solvers. In addition, by utilizing the collective intelligence nature of ACO, a new mechanism is proposed to shrink the problem scale as the optimization process iterates. Therefore, the proposed ACO-based tool-path optimizer can generate better solutions requiring shorter post-processing time comparing to the conventional ACO. Accuracy and strings problems were also investigated. It is shown that the proposed methods did not degrade the dimensional accuracy but significantly improved the visual quality of the printed parts by reducing the strings formed on the surface.

3. An efficient tool-path optimizer is proposed which was developed based on a new detour search algorithm and an efficient local search algorithm with new implementation by exploiting the unique properties in 3D printing applications. A detour search algorithm replaces unnecessary overheads on a path with detours which have relatively lower time cost. Besides, a k-opt local search algorithm has been utilized to solve many different types of routing or planning problems. However, when using a conventional implementation, its computational complexity increases rapidly with the parameter k or the scale of the problem. Based on some unique properties in 3D printing applications, it is mathematically proven that some combinations indicated by the proposed theorems can be neglected without degrading the quality of the solutions generated. The proposed tool-path optimizer has demonstrated its performance by accelerating the 3D printing process at most 36% with insignificant post-processing time in the test. Moreover, since the pro-

posed optimizer is an improvement algorithm. It is capable of cooperating with other algorithms to further improve the quality of solutions, such as the proposed ACO-based tool-path optimizer.

6.2 Suggestions for Future Work

In our previous study on using a machine learning approach to accelerate the optimization process [72], it is found that neural networks (NN) can be utilized to distinguish unfavorable transitions and be used to facilitate a local search algorithm.

Among existing TSP and URPP solvers, greedy algorithms are common and intuitive approaches since they have a relatively low computation complexities $O(n^2)$ when compared to other methods, where n is the size of a problem. Fig. 6.1(a) shows the optimal solution for the drilling problem "a280" [73]. Fig. 6.1(b) shows a tool-path generated using a greedy algorithm for the same problem. It can be observed that the cost (*i.e.* path length) of the tour generated using a greedy algorithm (3148 unit) is much higher than that of the optimum tour (2579 unit). In Fig. 6.1(b), there are several long tool-path motion segments (known as transitions) which lead to a relatively higher cost. To ease the explanation, the term *optimal transition* is utilized here to indicate a transition in the optimal tour.

The optimal transitions on the sub-optimal tour generated using a greedy algorithm are highlighted in Fig. 6.1(c). There are a total of 207 transitions that are regarded as optimal transitions, which is around 74% of all the transitions being generated. According to our preliminary study, similar results are obtained in other TSP instances.

If all the optimal transitions in a sub-optimal tour can be identified, the problem becomes finding a low-cost tour traversing all nodes and those identified

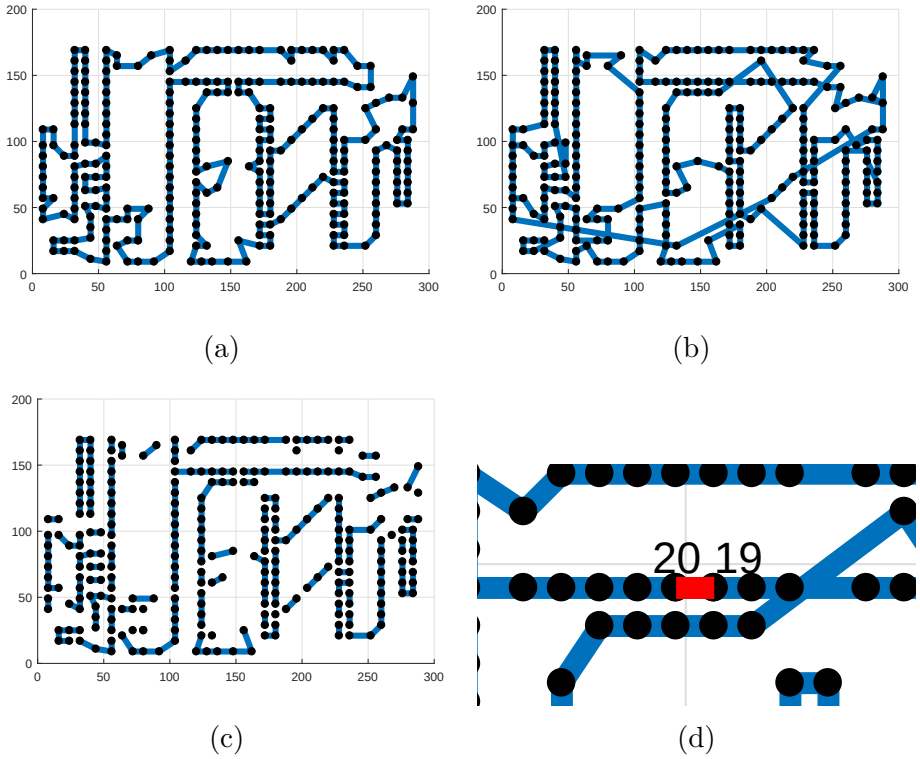


Figure 6.1: Illustrations showing (a) the optimum tour of the problem "a280" [73], (b) a tour generated using a greedy algorithm, (c) optimal transitions identified in (b), and (d) a non-optimal transition connecting nodes 19 and 20 generated from a greedy algorithm. Black dots and blue lines represent nodes and transitions, respectively.

optimal transitions, which will have a much lower computational complexity. However, identifying optimal transitions is a non-trivial task. There are no direct relations between the cost of a transition and the likelihood of it being one of the optimal transitions. For example, Fig. 6.1(d) highlights one transition from Fig. 6.1(b) which connects nodes 19 and 20. It can be observed that this transition is shorter than most of the transitions on the tour. However, it is not an optimal transition according to Fig. 6.1(a).

This study [72] revealed that machine learning approaches could be utilized to solve the tool-path planning problem. Recently, a framework is proposed [58] to solve combinatorial optimization problems using NN and deep reinforcement network (DQN). Their work utilized the point networks [74] which enable the

model to select a particular position in the input sequence instead of estimating the index of a fixed collection. Their work provided an interesting insight that DQN can be trained to solve TSP even though with TSP training data with a relatively small scale. A later study [75] improve the efficient of that framework. They found that a 2-opt local search algorithm worked properly with their proposed framework. Their results showed that the proposed method can deliver similar performances to other heuristics when solving TSP instances with a relatively smaller scale. Dai, Dai, and Song proposed a new approach [76], namely S2V, to represent structure data based on the idea of embedding latent variable models into feature spaces. Based on that approach, Dai *et al.* developed a generic framework [77], namely S2V-DQN, to solve a wide range of optimization problems over graphs such as TSP, minimum vertex cover problem, and maximum cut problem. The solutions generated by their method are on average 8.9% above the optimal solutions when utilized to solve the TSP instances not more than 1200 nodes.

The above studies demonstrated the potential of using DQN or other machine learning approaches to solve the tool-path optimization problem. A more in-depth study is needed to connect DQN and the tool-path optimization problem in 3D printing applications. For example, the above studies covered TSP instances with a relatively smaller scale ($n \leq 318$) where a printed part can be composed of half a million print segments. Further investigations are required to develop a practical tool-path optimizer which utilizes DQN to generate sub-optimal tool-paths in an efficient way.

Appendices

In a nozzle path planning problem, transitions $E_t = E \setminus E_r$ are classified exclusively into one of the following three subsets of E according to the following conditions.

$$E_A = \big\{e\colon e \in E \setminus E_r,\ t_R(e) \neq 0\big\},$$

$$E_B = \big\{e\colon e \in E \setminus E_r,\ t_R(e) = 0,\ d(e) > 0\big\},$$

$$E_C = \big\{e\colon e \in E \setminus E_r,\ t_R(e) = 0,\ d(e) = 0\big\}.$$

Here, $|E_A|$, $|E_B|$, and $|E_C|$ represent the number of transitions in the subset E_A, E_B, and E_C respectively. It is given that $t_D(d)$ represents the time cost required by a printing nozzle to traverse a transition with length d.

Lemma 1.

$$t_D(d_1 + d_2) \leq t_D(d_1) + t_D(d_2),$$

for all $d_1 \geq 0$ and $d_2 \geq 0$.

Proof.

Case 1: $d_1 \geq d_{\min}$, $d_2 < d_{\min}$, and $d_1 + d_2 \geq d_{\min}$.

$$\begin{aligned} t_D(d_1 + d_2) &= \frac{d_1 + d_2 + d_{\min}}{v_{\max}}. \\ t_D(d_1) + t_D(d_2) &= \frac{d_1 + d_{\min}}{v_{\max}} + \frac{2\sqrt{d_2 d_{\min}}}{v_{\max}} \\ &\geq \frac{d_1 + d_{\min} + 2d_2}{v_{\max}} \\ &\geq t_D(d_1 + d_2). \end{aligned}$$

Case 2: $d_1 \geq d_{\min}$, $d_2 \geq d_{\min}$, and $d_1 + d_2 \geq d_{\min}$.

$$
\begin{aligned}
t_D(d_1 + d_2) &= \frac{d_1 + d_2 + d_{\min}}{v_{\max}}. \\
t_D(d_1) + t_D(d_2) &= \frac{d_1 + d_2 + 2d_{\min}}{v_{\max}} \\
&\geq t_D(d_1 + d_2).
\end{aligned}
$$

Case 3: $d_1 < d_{\min}$, $d_2 < d_{\min}$, and $d_1 + d_2 < d_{\min}$.

$$
\begin{aligned}
t_D(d_1) + t_D(d_2) &= \frac{2\sqrt{d_{\min}}(\sqrt{d_1} + \sqrt{d_2})}{v_{\max}}. \\
t_D(d_1 + d_2) &= \frac{2\sqrt{d_{\min}}\sqrt{d_1 + d_2}}{v_{\max}} \\
&= \frac{2\sqrt{d_{\min}}\sqrt{(\sqrt{d_1} + \sqrt{d_2})^2 - 2\sqrt{d_1 d_2}}}{v_{\max}} \\
&\leq t_D(d_1) + t_D(d_2).
\end{aligned}
$$

Case 4: $d_1 < d_{\min}$, $d_2 < d_{\min}$, and $d_1 + d_2 \geq d_{\min}$.

$$
\begin{aligned}
t_D(d_1 + d_2) &= \frac{(d_1 + d_2) + d_{\min}}{v_{\max}}. \\
t_D(d_1) + t_D(d_2) &= \frac{2\sqrt{d_1 d_m}}{v_{\max}} + \frac{2\sqrt{d_2 d_m}}{v_{\max}} \\
&\geq \frac{2\sqrt{d_1 d_1}}{v_{\max}} + \frac{2\sqrt{d_2 d_2}}{v_{\max}} \\
&= \frac{d_1 + d_2 + (d_1 + d_2)}{v_{\max}} \\
&\geq t_D(d_1 + d_2).
\end{aligned}
$$

The Lemma 1. holds for all $d_1 \geq 0$ and $d_2 \geq 0$. ■

Lemma 2.

$$
t_D\big(\sum_{i=1}^{k}(d_i)\big) \leq \sum_{i=1}^{k} t_D(d_i),
$$

where $d_i \geq 0 \ \forall i \in [1, k]$ and $k \geq 2$.

Proof. The proof follows to that in Lemma 1, which is omitted here. ■

Theorem 1. The combinations of one E_B component and $(k-1)$ E_C components do not deliver improvement to the solution.

Proof. When apply a combination that selected one E_B transition and $(k-1)$ E_C components to a solution by using k-opt. The improvement can be indicated as

$$\Big(t_T(b) + \sum_{i=1}^{k-1} t_T(c_i)\Big) - \Big(\sum_{i=1}^{k} t_T(d_i)\Big), \tag{1}$$

where b is the length of the selected E_B transition and c_i is the length of the i-th selected E_C transitions. Here, d_i is the length of the i-th transitions that established after updating the solution using that combination. Since the cost of any transition in E_C is zero and any transition in E_B or E_C is not associated with retraction, (1) can be rewritten as

$$\Big(t_D(b)\Big) - \Big(\sum_{i=1}^{k} t_T(d_i)\Big), \tag{2}$$

Furthermore, by using the triangle inequality, it is known that

$$b \leq \sum_{i=0}^{k} d_i.$$

Since $t_D(d)$ is strictly increasing and continuous for $d \geq 0$, it can be shown that

$$\Big(t_D(b)\Big) \leq t_D\Big(\sum_{i=1}^{k}(d_i)\Big).$$

With Lemma 2,

$$\Big(t_D(b)\Big) \leq t_D\Big(\sum_{i=1}^{k}(d_i)\Big) \leq \sum_{i=1}^{k} t_D(d_i) \leq \sum_{i=1}^{k} t_T(d_i). \tag{3}$$

By considering (2) and (3),

$$\Big(t_D(b)\Big) - \Big(\sum_{i=1}^{k} t_T(d_i)\Big) \leq 0.$$

Theorem 1 holds for all $k \geq 2$. ■

Theorem 2. Combinations with only E_C components do not deliver improvement to the solution.

Proof. It is known that for all combinations with only E_C components, the improvement can be calculated as

$$\begin{aligned}\sum_{i=1}^{k} t_T(c_i) - \Big(\sum_{i=1}^{k} t_T(d_i)\Big) &= -\Big(\sum_{i=1}^{k} t_T(d_i)\Big) \\ &\leq 0\end{aligned}$$

Theorem 2 holds for all $k \geq 2$. ■

Bibliography

[1] C. R. (n.d.). 3D printing market size worldwide from 2013 to 2021 (in billion u.s. dollars). in statista - the statistics portal. (Accessed: 2019-03-18). [Online]. Available: https://www.statista.com/statistics/796237/worldwide-forecast-growth-3d-printing-market

[2] Crunchbase. List of top 3D printing companies. (Accessed: 2019-03-18). [Online]. Available: https://www.crunchbase.com/hub/3d-printing-companies

[3] "3D printing materials guide - comparing the 13 best filaments," (Accessed: 2019-03-18). [Online]. Available: https://www.simplify3d.com/support/materials-guide

[4] "WOODFILL," (Accessed: 2019-03-18). [Online]. Available: https://colorfabb.com/woodfill

[5] U. of Canterbury. Kiwi students create world-first 3D-printed titanium engine for eco-marathon car. (Accessed: 2019-03-18). [Online]. Available: https://www.canterbury.ac.nz/news/2018/kiwi-students-create-world-first-3d-printed-titanium-engine-for-eco-marathon-car.html

[6] AREVO. AREVO unveils first 3D-printed carbon-fiber ebike. (Accessed: 2019-03-18). [Online]. Available: https://arevo.com/news_item/arevo-unveils-first-3d-printed-carbon-fiber-ebik

[7] bigrep. World-first 3D printed airless bicycle tire. (Accessed: 2019-03-18). [Online]. Available: https://bigrep.com/posts/world-first-3d-printed-airless-bicycle-tire

[8] S. V. Murphy and A. Atala, "3D bioprinting of tissues and organs," *Nature biotechnology*, vol. 32, no. 8, p. 773, 2014.

[9] M. Tony, "Gold Coast man receives 3D-printed shinbone in world-first surgery," (Accessed: 2019-03-18). [Online]. Available: https://www.smh.com.au/national/queensland/gold-coast-man-receives-3d-printed-shinbone-in-world-first-surgery-20170908-p4yvsx.html

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

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

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

[13] C. Orloff, "A fundamental problem in vehicle routing," *Networks*, vol. 4, no. 1, pp. 35–64, 1974.

[14] J. K. Lenstra and A. Kan, "On general routing problems," *Networks*, vol. 6, no. 3, pp. 273–280, 1976.

[15] 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.

[16] Jeongho Son and Sunghee Choi, "Orientation selection for printing 3D models," in *2015 International Conference on 3D Imaging (IC3D)*, Dec 2015, pp. 1–6.

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

[18] W. Cheng, J. Fuh, A. Nee, Y. Wong, H. Loh, and T. Miyazawa, "Multi-objective optimization of part-building orientation in stereolithography," *Rapid Prototyping Journal*, vol. 1, no. 4, pp. 12–23, 1995.

[19] J. Zarbakhsh, A. Iravani, and Z. Amin-Akhlaghi, "Sub-modeling finite element analysis of 3D printed structures," in *2015 16th International Conference on Thermal, Mechanical and Multi-Physics Simulation and Experiments in Microelectronics and Microsystems*, April 2015, pp. 1–4.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] W. Wang, H. Chao, J. Tong, Z. Yang, X. Tong, H. Li, X. Liu, and L. Liu, "Saliency-preserving slicing optimization for effective 3D printing," in *Computer Graphics Forum*, vol. 34, no. 6. Wiley Online Library, 2015, pp. 148–160.

[26] B. Ezair, F. Massarwi, and G. Elber, "Orientation analysis of 3D objects toward minimal support volume in 3D-printing," *Computers & Graphics*, vol. 51, pp. 117–124, 2015.

[27] B. Thompson and H. Yoon, "Efficient path planning algorithm for additive manufacturing systems," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 4, no. 9, pp. 1555–1563, Sep. 2014.

[28] 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.

[29] W. Sheng, N. Xi, M. Song, and Y. Chen, "Robot path planning for dimensional measurement in automotive manufacturing," *Journal of manufacturing science and engineering*, vol. 127, no. 2, pp. 420–428, 2005.

[30] 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.

[31] G. S. Tewolde and W. Sheng, "Robot path integration in manufacturing processes: Genetic algorithm versus ant colony optimization," *IEEE Trans-*

actions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 38, no. 2, pp. 278–287, March 2008.

[32] W. Sheng, G. Tewolde, and H. Chen, "Tool path integration for spray forming processes using a genetic algorithm," in *ICAR'05. Proceedings., 12th International Conference on Advanced Robotics, 2005.* IEEE, 2005, pp. 159–164.

[33] C. H. Papadimitriou and M. Yannakakis, "The traveling salesman problem with distances one and two," *Mathematics of Operations Research*, vol. 18, no. 1, pp. 1–11, 1993.

[34] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," DTIC Document, Tech. Rep., 1976.

[35] G. N. Frederickson, "Approximation algorithms for some postman problems," *Journal of the ACM*, vol. 26, no. 3, pp. 538–554, Jul 1979.

[36] C.-T. Cheng, K. Fallahi, H. Leung, and C. K. Tse, "A genetic algorithm-inspired UUV path planner based on dynamic programming," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1128–1134, 2012.

[37] H. Delaram, A. Dastfan, and M. Norouzi, "Optimal thermal placement and loss estimation for power electronic modules," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 8, no. 2, pp. 236–243, 2018.

[38] A. Yahyaoui, N. Fnaiech, and F. Fnaiech, "A suitable initialization procedure for speeding a neural network job-shop scheduling," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 3, pp. 1052–1060, March 2011.

[39] M. Watanabe, M. Furukawa, A. Mizoe, and T. Watanabe, "GA applications to physical distribution scheduling problem," *IEEE Transactions on Industrial Electronics*, vol. 48, no. 4, pp. 724–730, Aug 2001.

[40] 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.

[41] 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.

[42] L. Da Xu, C. Wang, Z. Bi, and J. Yu, "Autoassem: an automated assembly planning system for complex products," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 3, pp. 669–678, 2012.

[43] A. F. Alkaya and E. Duman, "Application of sequence-dependent traveling salesman problem in printed circuit board assembly," *IEEE transactions on components, packaging and manufacturing technology*, vol. 3, no. 6, pp. 1063–1076, 2013.

[44] 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.

[45] 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.

[46] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 2. IEEE, July 1999, pp. 1470–1477 Vol. 2.

[47] M. Mavrovouniotis, F. M. Müller, and S. Yang, "Ant colony optimization with local search for dynamic traveling salesman problems," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1743–1756, 2017.

[48] F. Zheng, A. Zecchin, J. Newman, H. Maier, and G. Dandy, "An adaptive convergence-trajectory controlled ant colony optimization algorithm with application to water distribution system design problems," *IEEE Transactions on Evolutionary Computation*, March 2017.

[49] M. Mahi, Ö. K. Baykan, and H. Kodaz, "A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem," *Applied Soft Computing*, vol. 30, pp. 484–490, 2015.

[50] R. Skinderowicz, "The GPU-based parallel ant colony system," *Journal of Parallel and Distributed Computing*, vol. 98, pp. 48–60, 2016.

[51] G. A. Croes, "A method for solving traveling-salesman problems," *Operations research*, vol. 6, no. 6, pp. 791–812, Dec 1958.

[52] A. Hertz, G. Laporte, and P. N. Hugo, "Improvement procedures for the undirected rural postman problem," *INFORMS Journal on Computing*, vol. 11, no. 1, pp. 53–62, Feb 1999.

[53] L. Muyldermans, P. Beullens, D. Cattrysse, and D. Van Oudheusden, "Exploring variants of 2-opt and 3-opt for the general routing problem," *Operations research*, vol. 53, no. 6, pp. 982–995, Dec 2005.

[54] G. Groves and J. Van Vuuren, "Efficient heuristics for the rural postman problem," *ORiON*, vol. 21, no. 1, pp. 33–51, Jan 2005.

[55] Ş. Gülcü, M. Mahi, Ö. K. Baykan, and H. Kodaz, "A parallel cooperative hybrid method based on ant colony optimization and 3-opt algorithm for solving traveling salesman problem," *Soft Computing*, pp. 1–17, 2016.

[56] E. P. Ijjina and K. M. Chalavadi, "Human action recognition using genetic algorithms and convolutional neural networks," *Pattern recognition*, vol. 59, pp. 199–212, 2016.

[57] R. Zhang, J. Tao, and H. Zhou, "Fuzzy optimal energy management for fuel cell and supercapacitor systems using neural network based driving pattern recognition," *IEEE Transactions on Fuzzy Systems*, pp. 1–12, 2018 [Early Access].

[58] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *Learning Representations (ICLR), 2017 International Conference on*, 2017.

[59] N. Ganganath, C. T. Cheng, K. Y. Fok, and C. K. Tse, "Trajectory planning for 3D printing: A revisit to traveling salesman problem," in *2016 2nd International Conference on Control, Automation and Robotics (ICCAR)*, April 2016, pp. 287–290.

[60] T. Kuipers, "CuraEngine/tests at 4c547b9a66433885d4d4128f5f416982878b7e56 · Ultimaker/CuraEngine · GitHub," (Accessed: 2016-07-01). [Online]. Available: https://github.com/Ultimaker/CuraEngine/tree/4c547b9a66433885d4d4128f5f416982878b7e56/tests

[61] Ultimaker, "Cura 3D Printing Slicing Software," (Accessed: 2016-01-2). [Online]. Available: https://ultimaker.com/en/products/cura-software

[62] M. Dietz, "GCode Print Simulator by mdietz - Thingiverse," (Accessed: 2016-01-7). [Online]. Available: http://www.thingiverse.com/thing:44286

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

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

[65] N. Ganganath, C. Cheng, and C. K. Tse, "A constraint-aware heuristic path planner for finding energy-efficient paths on uneven terrains," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 601–611, June 2015.

[66] R. Flórez, "Hole Test by magramo - Thingiverse," (Accessed: 2018-01-11). [Online]. Available: http://www.thingiverse.com/thing:2380801

[67] K. Y. Fok, "GCodeAnalysor-1.0 by kyfok - Thingiverse," (Accessed: 2016-11-9). [Online]. Available: http://www.thingiverse.com/thing:1870254

[68] "Calibration Cube Collection by thingster - Thingiverse," (Accessed: 2018-01-11). [Online]. Available: http://www.thingiverse.com/thing:56671

[69] 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.

[70] "Popular Collection collection - Thingiverse," (Accessed: 2018-04-02). [Online]. Available: https://www.thingiverse.com/kyfok/collections/popular-collection

[71] K.-F. Man, K.-S. Tang, and S. Kwong, *Genetic algorithms: concepts and designs*. Springer Science & Business Media, 2012.

[72] K.-Y. Fok, N. Ganganath, C.-T. Cheng, H. H.-C. Iu, and C. K. Tse, "Toolpath optimization using neural networks," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2019, Manuscript accepted for publication.

[73] G. Reinelt, "TSPLIB–A traveling salesman problem library," *ORSA journal on computing*, vol. 3, no. 4, pp. 376–384, 1991.

[74] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2692–2700.

[75] M. Deudon, P. Cournut, A. Lacoste, Y. Adulyasak, and L.-M. Rousseau, "Learning heuristics for the TSP by policy gradient," in *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 2018, pp. 170–181.

[76] H. Dai, B. Dai, and L. Song, "Discriminative embeddings of latent variable models for structured data," in *International conference on machine learning*, 2016, pp. 2702–2711.

[77] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 6348–6358.