The following publication Fok, K. Y., Cheng, C. T., Ganganath, N., Iu, H. H. C., & Chi, K. T. (2018, May). Accelerating 3D Printing Process Using an Extended Ant Colony Optimization Algorithm. In 2018 IEEE International Symposium on Circuits and Systems (ISCAS) (pp. 1-5). IEEE is available at https://dx.doi.org/10.1109/ISCAS.2018.8351113

Accelerating 3D Printing Process Using an Extended Ant Colony Optimization Algorithm

Kai-Yin Fok[†*], Chi-Tsun Cheng[†], Nuwan Ganganath[‡], Herbert Ho-Ching Iu[‡], and Chi K. Tse[†]
[†]Dept. of Electronic and Information Eng., The Hong Kong Polytechnic University, Hung Hom, Hong Kong
[‡]School Electrical, Electronic and Computer Eng., The University of Western Australia, Crawley, WA, Australia
Email: *zerofky@gmail.com

***Abstract*—Ant colony optimization (ACO) algorithms have been widely adopted in solving combinatorial problems, like the traveling salesman problem (TSP). Nevertheless, with a proper transformation to TSP, ACO is capable of solving undirected rural postman problems (URPP) as well. In fact, nozzle path planning problems in 3D printing can be represented as URPP. Therefore, in this work, ACO is utilized as a URPP solver to accelerate the printing process in fused deposition modeling applications. Furthermore, mechanisms which exploit unique properties in 3D models are proposed to further extend the ACO in the above optimization process. These mechanisms are capable of accelerating ACO by adaptively adjusting its number of iterations on-the-fly. Simulation results using real-life 3D models show that the proposed extensions can accelerate ACO without affecting the quality of its solutions significantly.**

***Index Terms*—Ant colony optimization, Additive manufacturing, 3D printing, Undirected rural postman problem**

I. Introduction

Additive manufacturing is the terminology refers to the fabrication of 3D models by depositing materials in a layer by layer manner. A typical example of additive manufacturing is fused deposition modeling (FDM) which is more widely known as 3D printing. To print a computer-aided design (CAD) model, the model is first sliced into multiple thin layers using a slicer software. In each layer, the model is represented using large volumes of print segments. The segments are then converted into control codes by the slicer. A 3D printer is then instructed to perform the corresponding machining motions according to the control codes. When the nozzle traverses a print segment, its extruder injects filament toward the reservoir of the nozzle and causes the molten filament to be deposited to the desired location on a print bed. The printing nozzle traverses to the next print segments and repeats the process until all print segments on the current layer have been deposited. The print bed is then descended and the printing process of the next layer proceeds by depositing material on top of the layers underneath additively.

The build time of a model mainly comprises the time spent on depositing print segments and maneuvering a nozzle between segments known as transitions. The time spent on depositing material is restricted by the specifications of a 3D printer, such as its filament flow rate and the rating of its heating element. However, the time spent on traversing between print segments can be shortened by carefully planning the path of the nozzle. In this work, the routing problem of the printing nozzle is formulated as an undirected rural postman problem (URPP) [1]. Given a set of edges E (*i.e.* print segments and transitions) and a set of required edges $E_r \in E$ (*i.e.* print segments only), the objective is to find a fast path that traverses all E_r at least once. URPP is proven to be NP-hard if the subgraph induced by the required edges are not connected [2].

Researches have been carried out to improve the performance of 3D printing applications in many different aspects. Wang *et al.* [3] studied the visual quality of printed objects. They proposed an adaptive slicing scheme which is able to adjust layer thickness according to the structures of 3D models. They showed that the corresponding build time can be reduced by using their method while a high visual quality can still be preserved. Ezair *et al.* [4] studied the generation of supporting structures in 3D printing. In order to minimize the amount of materials used to form the supporting structures, they proposed an algorithm which can select an appropriate orientation according to the model to be built. In [5], Fok *et al.* proposed a print plan optimizer based on a deterministic method by which sub-optimum solutions were obtained with a reasonable computational cost. Recently, they formulated the nozzle path planning problem in 3D printing applications into a URPP [6]. They modified an existing URPP solver which successfully shorten the build time by eliminating unnecessary movements of the printing nozzle.

An ant colony optimization (ACO) algorithm is utilized in this work as a path optimizer for 3D printing applications. ACO is a nature-inspired meta-heuristic that was proposed by Dorigo [7] and first used to solve traveling salesmen problems (TSPs). Nowadays, ACO is applied to solve many other combinatorial optimization problems such as path planning and networking problems [8]–[10]. Extensive researches have been conducted to improve the performance of ACO. Zheng *et al.* [11] proposed a parameter-adaptive strategy for ACO. In their work, control parameters can be adjusted dynamically during the optimization process. Mahia *et al.* [12] utilized a particle swarm optimization (PSO) method to optimize the parameters in ACO according to the quality of solutions generated by the artificial ants. Skinderowicz [13] proposed several parallel versions of ACO which utilized graphics processing units (GPUs).

In this work, two mechanisms are proposed to extend ACO for 3D printing applications. The proposed mechanisms ac-

celerate ACO by adaptively adjusting its number of iterations on-the-fly according to the unique properties in the given sliced 3D models. The rest of the paper is arranged as follow. Section II provides the problem formulation. The proposed mechanisms are developed based on unique properties found in 3D printing. Details of the properties and the corresponding mechanisms are elaborated in Section III. Performances of the extended ACO were evaluated using computer simulations. Settings and discussions regarding the simulations are presented in Section IV followed by the simulation results. Finally, Section V concludes the whole paper.

II. Problem formulation

In this work, the printing nozzle path planning problem in each layer of a sliced model is formulated as a URPP. An overall optimized print plan can later be obtained by connecting the optimized paths on different layers successively. In a URPP, a connected and undirected graph is denoted as $G = (V, E)$. The aim is to find a fast tour, which can visit all the required edges $E_{\mathrm{r}} \in E$. Under the URPP formulation, print segments are regarded as the required edges E_{r}. All vertices corresponding to edges in E_{r} form a vertex set V. The superset E is formed by including other edges that connecting every vertex pair in V. The set of transitions can therefore be denoted as $E \setminus E_{\mathrm{r}}$. The objective is to find a fast tour which traverses all E_{r} on G. Since the total time required by the printing nozzle to traverse all the print segments in the model is a constant, during the optimization process, only the cost associated with transitions will be considered. In this work, a valid solution represents a route traversing all print segments. The fitness value of a solution is denoted as the time required for the printing nozzle to traverse all the associated transitions on a valid solution.

A. Ant Colony Optimization

ACO has been widely used to solve TSP, which artificial ants are used to find routes that traverse all required vertices. Information exchanges among ants are achieved via depositing artificial pheromone on the graph. In [14], it has been demonstrated that ACO is also capable of solving URPP by first transforming the problem into TSP. In this work, the printing nozzle path planning problem is first formulated as a URPP, then transformed to TSP, and finally optimized using ACO.

The amount of pheromone deposited on an edge is inversely proportional to the cost of the intermediate routing solutions. Therefore, at the end of each iteration, a pheromone matrix is updated based on the solutions generated by the ants. Here, $\tau_{i,j}$ represents the pheromone level on an edge (i, j), where i and j are the two endpoints of that edge. Apart from pheromone information, a heuristic information $\eta_{i,j}$, is also imposed on edge (i, j), which is inversely proportional to the cost of that edge. When an ant searches iteratively for a solution, both heuristic and pheromone information are considered. Assuming the k^{th} ant is now at vertex i, the probability for it to choose (i, j) in its next step is expressed as [8]

$$p_{i,j}^{k}(t) = \frac{[\tau_{i,j}(t)]^{\alpha}[\eta_{i,j}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{i,l}(t)]^{\alpha}[\eta_{i,l}]^{\beta}}. \quad (1)$$

Here, N_i^k is a set of vertices that have valid paths with vertex i and are not yet visited by the k^{th} ant. The parameters α and β are for controlling the behavior of the ants. A detail study on the selection of α and β can be found in [12].

To avoid local optimum points and premature convergences, an operation called evaporation is used in ACO. It reduces pheromone concentrations of all edges proportionally at the end of each iteration. Therefore, at the end of the t^{th} iteration, the pheromone level $\tau_{i,j}$ on the edge (i, j) is updated as

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

Here, m denotes the number of ants utilized in the ACO. Nevertheless, $\Delta\tau_{i,j}^{k}(t)$ represents the amount of pheromone deposits by the k^{th} ant on edge (i, j) in the t^{th} iteration. The pheromone evaporation rate is controlled by $\rho \in (0, 1)$. While the search process continues iteratively, pheromone on edges included in poor solutions will receive significant evaporation and will be eliminated in long run.

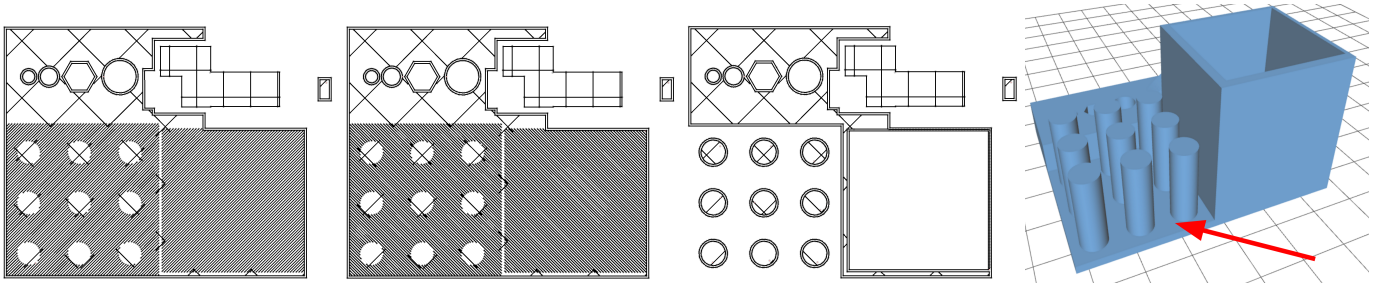
III. Proposed method

In this paper, ACO is used to optimize print plans of 3D models by reducing their corresponding total transition length such that their build times can be shortened. An extended ACO is proposed in this work, specifically for 3D printing applications, which exploits unique properties in typical sliced 3D models. In this section, the properties and the corresponding mechanisms are introduced.

As mentioned before, a CAD model is first decomposed into many thin layers. In order to print the model with high visual quality, accuracy, and fidelity, layers have to be thin. As an example, the commercial slicing software in [15] has its default layer thickness equals 0.1 mm. Thinner layers are sometimes required to build models with finer details. Under such conditions, any two adjacent layers of a sliced model usually share similar structures as the shape of a 3D model normally changes gradually. Figs. 1(a) and 1(b) illustrate two adjacent layers of a model "TortureTestV2" [16].

According to Figs. 1(a) and 1(b), it can be observed that locations and orientations of print segments on these two adjacent layers are quite similar. Shapes in the two layers are highly comparable except the orientations of some infilling lines. The proposed method utilizes this property, which adjacent layers usually have similar structures, to extend generic ACO for optimizing print plans.

To ease reading, some terminologies used in this work are introduced as follows. In ACO, ants search iteratively for a fast route for the nozzle to traverse all print segments. The number of iterations to be executed, namely N_{ite}, is a tuning parameter which is defined by the user and its value is normally fixed throughout the whole optimization process. ACO uses a stochastic mechanism to search for improvements. It is possible that no further improvement can be found in some of the iterations. For convenience, the term *effective iteration* is introduced in this paper to indicate an iteration in which possible improvements can be found. When



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

Fig. 1: Illustrations showing (a) the 16$^{\text{th}}$ layer, (b) the 17$^{\text{th}}$ layer, (c) the 18$^{\text{th}}$ layer, and (d) the highlighted CAD of the same model "TortureTestV2" [16] for which the surface formed on the 17$^{\text{th}}$ layer is indicated on the CAD.

optimizing the print plan of a sliced model using ACO, the total number of effective iterations on its i^{th} layer is expressed as $N^i_{\text{eff}} \in [0, N_{\text{ite}}]$.

Empirical studies were conducted on different 3D models to investigate the number of effective iterations across layers, which suggested that N^i_{eff} approximates to $N^{(i-1)}_{\text{eff}}$ for most of the layers among the models under test. Furthermore, the results also showed that ACO usually cannot achieve further improvement on the i^{th} layer when $N^i_{\text{eff}} > N^{(i-1)}_{\text{eff}}$. One possible reason is that the $(i-1)^{\text{th}}$ and the i^{th} layers are normally very alike, thus ACO requires similar numbers of iterations to explore the searching spaces. In this work, the proposed method utilizes the above properties to adaptively adjust the number of iterations in ACO utilized in the optimization processes.

Nevertheless, there are some exceptional cases that cause N^i_{eff} to differ from $N^{(i-1)}_{\text{eff}}$ significantly. When comparing the 17$^{\text{th}}$ and the 18$^{\text{th}}$ layers of the model "TortureTestV2", which are shown in Figs. 1(b) and Fig. 1(c) respectively, it can observed that they have relatively high deviations in terms of volume and density of print segments. These two layers have significantly different structures as some print segments on the 17$^{\text{th}}$ layer are used for constructing part of the model's surface, which is indicated by the red arrow in Fig. 1(d). Since no print segments will be deposited on top of the surface region on the 17$^{\text{th}}$ layer, as a result, N^{17}_{eff} differs from N^{18}_{eff} significantly. To make the optimization process more efficient and effective, the numbers of iterations required for these two layers should, therefore, be different. In this work, two mechanisms are proposed to adjust the number of iterations of each layer adaptively which are elaborated as follows.

A. Early Termination Mechanism

As mentioned before, two adjacent layers usually have similar structures. Therefore it can be expected that N^i_{eff} and $N^{(i-1)}_{\text{eff}}$ are similar. In the first mechanism, when ACO is optimizing the i^{th} layer, if the current number of effective iterations is greater than that of the $(i-1)^{\text{th}}$ layer, *i.e.* $N^{(i-1)}_{\text{eff}} > N^i_{\text{eff}}$, it is very likely that no more improvement can be found in the next iteration, and therefore, the optimization process on the current layer will be terminated. On the contrary, it is possible for two adjacent layers to have the different structures. To address these exceptional cases, even when $N^i_{\text{eff}} > N^{(i-1)}_{\text{eff}}$, if an improvement is recorded in the current iteration, the optimization process will continue until no further improvements can be found or some stopping criteria have been reached. The stopping criteria and the corresponding mechanism are explained in the next section.

B. Reallocation of Truncated Iterations

If the optimization process on the i^{th} layer terminated early, *i.e.* $N^i_{\text{ite}} < N_{\text{ite}}$, its remaining number of iterations is then stored in a global reserve N_{res}. In generic ACO, the optimization process is terminated if the number of iterations reaches N_{ite} regardless of any potential improvement. In the proposed method, the optimization process of the i^{th} layer is allowed to exceed N_{ite} if it has improvement in the current iteration and $N_{\text{res}} > 0$. Every extra iteration beyond N_{ite} will be compensated by those surplus iterations stored in N_{res}. The rationale of such design is to have a better allocation of iteration numbers across ACO operations in different layers of the 3D models. Therefore, layers with lower complexities can be terminated earlier, the surplus computational power can then be redistributed to layers that require more extensive searches. To give a fair comparison of the extended ACO and generic ACO, in this work, both methods are provided with the same N_{ite}. Furthermore, N_{res} in the proposed method was initialized to 0 at the beginning of the optimization process.

IV. Simulations

A. Simulation Settings

Simulations were performed to evaluate the performance of the proposed ACO in speeding up 3D printing processes. In the simulations, a commercial slicer software Cura [15] was used to slice CAD models into layers. The slicer software generated initial plans using its build-in greedy-based optimizer. In this work, a filling density of 10% was used for Cura to generate the initial print plans of the models. The methods under test were then applied to optimize the initial print plans. To give a fair comparison, 7 models in [16] with different unique properties were chosen as benchmarking models.

TABLE I: ESTIMATED BUILD TIME (s) OF PRINT PLANS OBTAINED USING DIFFERENT OPTIMIZERS.

Models	Cura	ACO				Extended ACO			
		Mean	Max	Min	SD	Mean	Max	Min	SD
3DHackerTest	6551.09	5421.05	5431.65	5412.64	5.16	5418.07	5433.83	5403.67	9.89
ctrlV_3D_test	4620.90	4095.21	4115.61	4075.04	14.77	4093.67	4106.78	4076.74	8.15
Debailey_x10	5263.32	4725.00	4745.07	4703.78	12.03	4725.65	4759.24	4697.79	22.73
dragon_65_tilted_large	3812.74	3083.51	3099.22	3060.56	10.64	3086.78	3099.42	3076.83	7.58
testModel	2537.31	2470.38	2480.27	2461.47	6.48	2479.86	2491.55	2469.29	7.78
TortureTestV2	9951.90	8835.38	8842.75	8829.67	4.32	8835.07	8845.25	8815.06	8.77
UltimakerRobot_support_2015	1829.68	1663.11	1670.60	1645.64	7.09	1664.16	1675.37	1655.34	5.05

TABLE II: POST-PROCESSING TIME (s) OF PRINT PLANS OBTAINED USING DIFFERENT OPTIMIZERS.

Models	ACO	Extended ACO
3DHackerTest	465.73	456.94
ctrlV_3D_test	170.73	159.12
Debailey_x10	194.91	173.50
dragon_65_tilted_large	211.85	199.15
testModel	27.61	24.38
TortureTestV2	221.47	204.51
UltimakerRobot_support_2015	122.96	107.74

To evaluate the performance of the proposed method, a generic ACO with a constant iteration number was also included in the tests. Both methods were implemented on a GPU-accelerated TSP solver [13] via a URPP to TSP transformation. In both optimizers, the values of α, β, and ρ were chosen as 1, 3, and 0.2 respectively as in [13]. Furthermore, $N_{\text{ite}} = 100$ for all the layers. While in the extended ACO, the actual utilized number of iterations in each layer was adjusted by the two proposed mechanisms adaptively on-the-fly.

The optimized plans were evaluated using an open-source 3D printing simulator [17]. Each method under test was used to optimize each model 10 times to obtain the corresponding average estimate build time and post-processing time, which is the time required to further optimize a given print plan. The simulations were conducted on a computer with Intel Core i7 processors, 16 GB RAM, GeForce GT 750M, Ubuntu 16.04, and CUDA 8.0.61. Simulation results are shown in TABLE I and TABLE II.

B. Results and Discussion

According to TABLE I, the generic ACO and the extended ACO can always yield solutions with shorter estimated build time when compared with those obtained from Cura. In general, the generic ACO and the extended ACO can on average reduce 11.56% and 11.50% of the estimated build time respectively when comparing to the initial plan generated by Cura.

The generic ACO and the extended ACO can reduce the estimated build time of the model "3DHackerTest" by 17.25% and 17.30% respectively at maximum. One possible reason is that there is a larger number of print segments and irregular shapes in such model when comparing with the others. Therefore, there are more rooms for ACO to optimize the print plan. Furthermore, the generic ACO and the extended ACO can both deliver similar performances on reducing the estimated build time of all the 7 models.

According to TABLE II, the extended ACO required shorter post-processing time when compared with the generic ACO. The proposed method required on average 8.20% less time than that of the generic ACO. Nevertheless, the proposed method can at maximum save 12.38% of post-processing time comparing to the generic ACO in optimizing the model "UltimakerRobot_support_2015". One possible reason is that the structure of such model is quite uniform across its layers, such that, the proposed method was capable to adjust the number of iterations effectively without degrading its solution quality. On the other hand, such improvement gap was reduced to 1.87% for model "3DHackerTest". As mentioned before, such model is populated with irregular shapes. The proposed method can still adaptively reallocate iterations across layers even when the structures of adjacent layers are having significant deviations by adopting the mechanism mentioned in Section III.

V. CONCLUSION

In this paper, an extended version of ACO for accelerating 3D printing applications is presented. By exploiting the unique properties between adjacent layers in sliced CAD models, the proposed method can adaptively adjust the number of iterations in ACO on-the-fly. Computer simulations were conducted to evaluate its performances. Simulation results show that the proposed method can significantly accelerate both the optimization and printing processes without compromising the quality of solutions.

ACKNOWLEDGMENT

This work is supported by the Department of Electronic and Information Engineering, the Hong Kong Polytechnic University (Projects RU9D and G-YBXK).

REFERENCES

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

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

[3] 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.

[4] 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.

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

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

[7] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," in *Proceedings of the 1999 Congress on Evolutionary Computation (CEC99)*, vol. 2. IEEE, 1999, pp. 1470–1477.

[8] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[9] N. Ganganath and C.-T. Cheng, "A 2-dimensional ACO-based path planner for off-line robot path planning," in *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC 2013)*. IEEE, 2013, pp. 302–307.

[10] N. Ganganath, C.-T. Cheng, and C. K. Tse, "An ACO-based off-line path planner for nonholonomic mobile robots," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS 2014)*. IEEE, 2014, pp. 1038–1041.

[11] 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.

[12] 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.

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

[14] M.-L. Pérez-Delgado, "Solving an arc-routing problem using artificial ants with a graph transformation," *Advances in Practical Applications of Agents and Multiagent Systems*, pp. 241–246, 2010.

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

[16] Ultimaker, "GitHub - Ultimaker/CuraEngine," (Accessed: 2017-02-22). [Online]. Available: https://github.com/Ultimaker/CuraEngine

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