Computational Design of Scheduling Strategies for Multi-Robot Cooperative 3D Printing

Conference Paper · August 2019

DOI: 10.1115/DETC2019-97640

3 authors, including:

Laxmi Prasad Poudel
University of Arkansas
10 PUBLICATIONS **13** CITATIONS

SEE PROFILE

Zhenghui Sha
University of Arkansas
55 PUBLICATIONS **212** CITATIONS

SEE PROFILE

Proceedings of the ASME 2019
International Design Engineering Technical Conferences
and Computers and Information in Engineering Conference
IDETC/CIE2019
August 18-21, 2019, Anaheim, CA, USA

IDETC2019-97640

COMPUTATIONAL DESIGN OF SCHEDULING STRATEGIES FOR MULTI-ROBOT COOPERATIVE 3D PRINTING

Laxmi Poudel
Department of Mechanical Engineering
University of Arkansas
Fayetteville, AR

Wenchao Zhou
Department of Mechanical Engineering
University of Arkansas
Fayetteville, AR

Zhenghui Sha[1]
Department of Mechanical Engineering
University of Arkansas
Fayetteville, AR

ABSTRACT

Cooperative 3D printing (C3DP) is a novel approach to additive manufacturing, where multiple printhead-carrying mobile robots work together cooperatively to print a desired part. The core of C3DP is the chunk-based printing strategy in which the desired part is first split into smaller chunks, and then the chunks are assigned to individual printing robots. These robots will work on the chunks simultaneously and in a scheduled sequence until the entire part is complete. Though promising, C3DP lacks proper framework that enables automatic chunking and scheduling given the available number of robots. In this study, we develop a computational framework that can automatically generate print schedule for specified number of chunks. The framework contains 1) a random generator that creates random print schedule using adjacency matrix which represents directed dependency tree (DDT) structure of chunks; 2) a set of geometric constraints against which the randomly generated schedules will be checked for validation; and 3) a printing time evaluation metric for comparing the performance of all valid schedules. With the developed framework, we present a case study by printing a large rectangular plate which has dimensions beyond what traditional desktop printers can print. The study showcases that our computation framework can successfully generate a variety of scheduling strategies for collision-free C3DP without any human interventions.

Keywords: Cooperative 3D printing, Task scheduling, Multirobot system

NOMENCLATURE

$AS_{R,i}(t)$	Accessible space of robot, *i,* at time, *t*
$SV_{R,i}(t)$	Swept volume of robot, *i,* at time, *t*
$AS_c\ (t)$	Occupied space by printed chunk at time, *t*

1. INTRODUCTION

As additive manufacturing (AM) transitions from rapid prototyping to digital manufacturing over the past years, the current cost structure of the technologies is too expensive to be viable for mainstream manufacturing adoption. One critical factor of the cost structure for adoption is the scalability issue, in terms of both print size and printing speed. Extensive research has been performed on increasing the size of the printer for printing larger parts, e.g., BAAM (big area additive manufacturing) system developed by Oak Ridge National Lab [1] and Sciacky EBAM 300 machine. Also, there are other researches on improving printing speed with multiple extruders. For example, Project Escher makes the use of multiple 3D print heads for massive jobs, where each print head acts like a separate printer but works in parallel on different areas of the same part [2]. It was demonstrated that the printing time can be significantly shortened with five extruders. Although the build volume is large, it is still limited to the size of the printer. Similarly, Jin et al. presented a concept of concurrent fused filament deposition, where multiple printing extruders are utilized simultaneously to print individual layer of a desired part [3]. They have developed an optimization model to minimize the printing makespan and developed a toolpath allocation and scheduling methodology for multiple extruders, where they allocate portion of each layer to individual extruders. They were able to reduce each layer printing time by as much as 60% using three extruders. While promising, the issue of scalability is not properly addressed as the demonstration has only been done using few extruders. Since the printing takes place in an enclosed box, this approach faces similar limitation of print size as Project Escher. Therefore, there is a limit on the maximum number of extruders that can be used in this system as well as the limit on the size of a part that can fit in the build volume.

[1] Contact author: zsha@uark.edu

While some of the aforementioned approaches made good progress on tackling the problem of print time and print size, the print quality might be impacted. A larger printer with large extruders can shorten the print time and accommodate larger part but the print quality will lower as a result of coarser resolution. In order to achieve a balance for this "impossible triangle" of print quality, print time, and print size, we have developed the Swarm 3D Printing and Assembly (SPA) platform, where a swarm of printhead-carrying mobile robots work simultaneously to print and assemble large objects [4]. One of the most important features of SPA, that distinguishes itself from traditional layer-based 3D printing system is the chunk-based printing strategy – a 3D model of a desired part is divided into smaller chunks first and each of these chunks are assigned to individual printing robots. Each individual robot prints one chunk at a time, but many printing robots work in parallel, and layer by layer for each chunk as illustrated in the **Figure 1**. Therefore, a large number of printing robots can be employed to print a large part. Since the printing can be parallelized, i.e., multiple chunks can be printed simultaneously, the total print time can be significantly reduced.

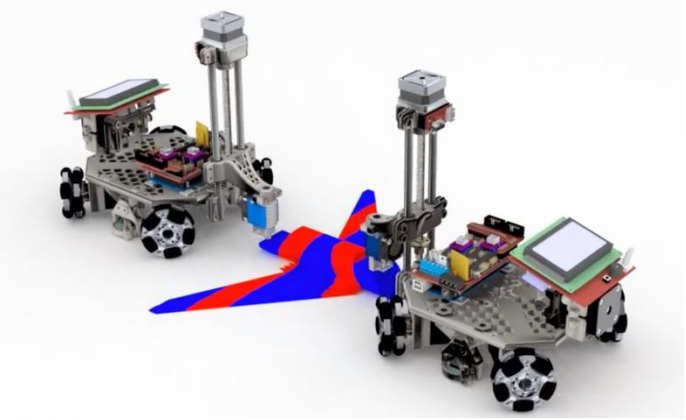

***Figure 1**: Chunk-based Cooperative 3D printing*

In our preliminary work, we have developed a chunk-based slicer in order to slice STL objects so that two 3D printing robots can work on printing the part simultaneously to reduce the total print time [5]. To achieve cooperative 3D printing (C3DP), the part is split into chunks with a sloped interface between them and the mechanical properties of the chunk-based parts studied in our prior work [6] shows that the chunk-based 3D printed part has comparable tensile strength to traditional layer-based 3D printed part. To scale C3DP to multiple robots in [7], we developed a heuristic-based scaling strategy, Scalable Parallel Array of Robots for 3DP (SPAR3), which enables a large number of mobile 3D printers to work cooperatively to finish a printing job without colliding with each other, as illustrated in **Figure 2**. First, two robots start working on the alternate chunks in the center row as shown in **Figure 2**. These robots then move over to print the remaining chunks in the same row in order to fill the gaps between the initially printed chunks. Once complete, active robots retreat to work on the next row of chunks. Meanwhile, the two additional robots, waiting for the completion of the central row of chunks, become active and start printing second row of chunks on the opposite side of central row as shown in the **Figure 2**. Similar to the central row, robots print alternate chunks first and once complete, move over to print the remaining chunks filling the gaps between previously printed chunks. This process alternates and continues until the entire part is finished.

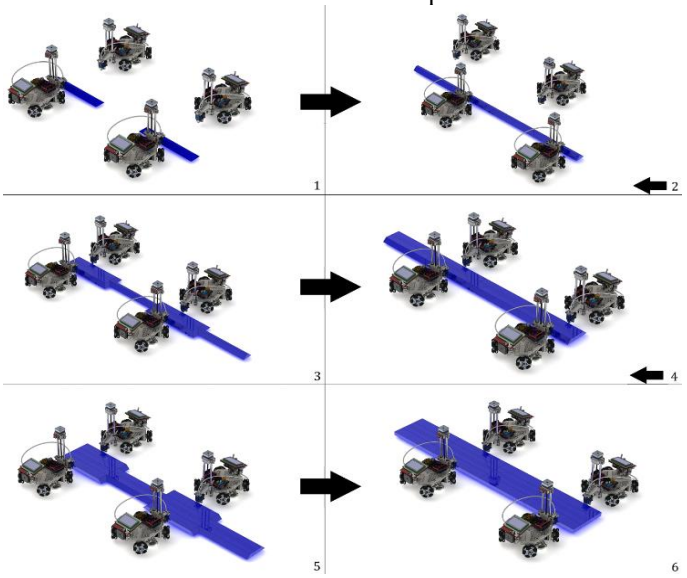

***Figure 2**: SPAR3 strategy illustration*

While a print schedule based on this simple heuristic for a simple geometry with finite number of robots might give us good results, as the number of chunks increases, the heuristics might not provide the best solution as the large part of design space remain unexplored. For example, if we increase the number of chunks from 20 to 100, ignoring the constraints, the possible number of print schedule increases exponentially. This would result in 100! possible plans just for sequential printing. Adding parallel printing (multiple chunks being printed simultaneously), increases the number of potential plans even more. So, we need to ensure that these larger design space unexplored by the heuristic approach is traversed thoroughly. In order to do so, a new approach needs to be developed so that, we can search for print schedules for complex geometries, which often cannot be handled by human heuristics, that can maximize the use of the resources in hand. A computational framework that can be transformed in algorithms so as to automatically generate many different valid scheduling strategies is essential to the development of scalable C3DP. Therefore, the **research objective** of this paper is to establish a computational framework that can explore larger design space that would otherwise remain unexplored and generate valid scheduling strategies from that space. In order to do just that, in this paper, we develop a new algorithm to automatically generate printing schedules for a given number of robots and evaluate the validity of the generated schedules based on the geometric constraints produced by both printing robots and chunks. Once validated, we can make comparison between the printing schedules heuristically generated with the ones computationally generated, using the printing time evaluation metrics.

The remaining of the paper is organized as follows. Section 2 presents the related work and the research gap that this paper aims to fill. Section 3 explains the general research approach and the computational framework, which includes the model for C3DP, geometric constraints and the time evaluation metrics. A case study is presented in Section 4 that demonstrate the utility and performance of the proposed framework. Finally, result of the case study and the discussion are presented in Section 5, followed by conclusion in Section 6.

2. RELATED WORK AND RESEARCH GAP

Although the SPA platform presents many benefits, as a new approach to digital manufacturing, it brings additional challenges in task scheduling (i.e., to optimally generate a printing sequence that minimizes the total time of printing, in our case) and task allocation (i.e., to assign chunks to individual robots). This makes it an integrated planning and scheduling (IPPS) problem. The addition of 3D printing as a manufacturing process compounds the difficulty of IPPS problem further, making it a NP-hard problem to solve [8][9]. IPPS has widely been researched ever since Chryssoulouris et al. first presented the concept of integrating the process planning and scheduling problems [10][11]. Though many optimization methods (simultaneously optimize both planning and scheduling functions) have been presented over the years using different approaches such as meta heuristics approach (e.g., Genetic Algorithm [11][12][14]) and agent based approaches (e.g., Particle Swarm Optimization [9] and Ant Colony Optimization [15]), the problem definition is limited around the job shop scheduling problems, where multiple jobs needs to be scheduled in multiple workstations. Scheduling plans for robotic applications is not studied in most of the related research. Robotic application adds more complexity to the problem as it necessitates the generation of motion trajectories for multiple mobile robots. Petrovic et al. performed a pioneering study to optimize schedule plans using chaos theory with particle swarm optimization (cPSO) algorithm and used it to generate motion trajectories followed by mobile robots in IPPS problem[9]. Though the work presents and demonstrates the motion trajectories of mobile robots, scope of work does not include collision detection between the mobile robots, where spatiotemporal constraints needs to be developed and applied.

Similarly, multi-robot systems (MRS) is another active field that deals with task assignment and collision-free scheduling of multiple robots. Koes et al. presented a novel framework and a centralized anytime algorithm with error bounds to address multi-robot scheduling and task allocation problem as mixed integer linear programming (MILP) problem, which could outperform greedy heuristics, and market-based approaches which separates scheduling and task allocation [16]. In a related study, a connectivity graph along with Liaison method was used to generate sequence for multi-robot assembly by Mishra et al. [17]. Parallel execution of assembly sequences for multi-robot work cell was studied by Park et al. [18]. While they developed constraints to avoid infeasible assembly sequence, no constraints were developed in order to avoid robot-to-robot collision as only single gripping robot was used for demonstration. A novel algorithm, Terico, was developed to generate schedule fulfilling temporospatial constraints for multi-robot system by Gombolay et al. [19]. This algorithm could perform near-optimal task assignments and schedule up to 10 robots and 500 tasks in less than 20 seconds on average but lacks clarity on how the algorithm behave as the number of robots is greater than 10. Wan et al. proposed a newly developed planner for finding optimal assembly sequence to assemble objects and demonstrated the results by optimally scheduling Soma cube [20]. While the scheduling approach of this work could be applicable, the work is only applicable to single robots, which eliminates complexity associated with the collision between working robots.

All of the literature discussed above, both in the field of IPPS and MRS, and other extant literatures provide different elegant solutions to multi-robot planning and scheduling problem and though at first glance it might seem like the problems addressed are similar to the problems facing the SPA platform, it is not the case. The literature in MRS are mostly focused on solving discrete problems, i.e., problems that can be solved by taking discrete number of steps, e.g., pick and place assembly, patterns formations, search and rescue, etc. While the literature in IPPS are mostly focused on job shop type of problems, where multiple machines (usually fixed) or workstations are available and multiple tasks needs to be completed. The problem of C3DP, on the other hand, is a unique blend of these two types of aforementioned problems. This complicates the problem, as not only do we have to worry about the planning and scheduling of chunks but also collisions between the mobile robots and their motion planning while doing so. In addition to this, the integration of 3D printing makes the manufacturing process continuous, where material is continuously deposited temporospatially until a desired part is manufactured. There is *no* existing method that could take a design of a desired part computationally and implement 3D printing using multiple printing robots. Thus, there is a clear gap for generating viable and valid scheduling strategies for C3DP. Moreover, different chunking methods of C3DP will result in different temporospatial constraints that could be even more complicated than our previously proposed slope angle-based chunking. Thus, a general framework is needed to handle parallel task scheduling for continuous 3D printing process with complex temporospatial constraints.

3. RESEARCH APPROACH

3.1 The Computational Framework for C3DP Scheduling

Prior to jumping at explaining the computational framework, it is paramount that we explain the entire printing process of C3DP. We approach the continuous problem of C3DP by first discretizing the entire process using a chunk-based approach. Doing so converts the continuous C3DP into multi-stage discrete process as illustrated in **Figure 3** such that there are inter-dependencies between multiple stages.

In order to achieve C3DP, first, a part is divided into small chunks. Once the chunk division is complete, a print schedule is generated for a given number of printing mobile robots. The scheduling is represented using a Directed Dependency Tree (DDT) where the nodes represent the chunks and the edges represent dependency between the nodes as shown in **Figure 4** [7]. A DDT can define both print schedule and the dependency relationships between the chunks at the same time. The order of layers from top to bottom represents the print order, and the number of chunks at each layer represents the number of print tasks that can be done in parallel.

In the proposed computational framework, a random DDT is first generated, which contains the scheduling information such as chunk dependencies and number of sequence (i.e., the depth of a tree). Then, the path planning is done for the generated DDT. Prior to printing, geometric constraints validation will be performed. The geometric constraints validation will ensure that no collision takes place between the printing robots (R2R collision) and between the printing robots and printed parts (R2P collision). This entire process of C3DP and the computational framework of automatically scheduling are depicted in **Figure 3**.

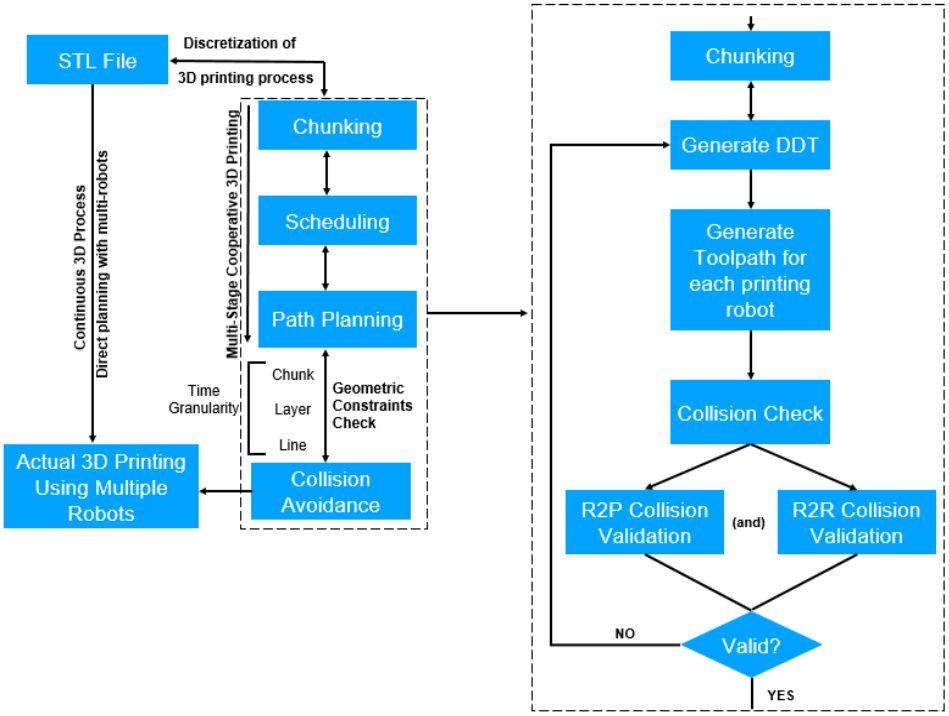

***Figure 3**: Flowchart depicting the different stages of C3DP after discretizing*

3.2 Random Generation of Print Sequence

For random generation of C3DP schedules, a part is first divided into n chunks by a chunker. Once the chunking is complete, a sparse adjacency matrix of dimension $n \times n$ is generated with binary values of either 1 or 0 at random indices. Value of 1 represents a dependent relationship between two chunks and 0 represents absence of dependency between the chunks. For example, if a part is divided into 12 chunks (numbered 0-11), one of the generated dependency matrices (and the corresponding dependency tree) might look like the one shown in **Figure 4**. Both the adjacency matrix and the DDT present the same information but only represented differently. The chunks with no dependency in the matrix will be placed at the root node at the top of the dependency tree (chunk 0 and chunk 1). The chunks that depend on either one of those chunks (or both) will be placed below the root nodes (chunk 2 and chunk3). This method of adding chunks as nodes is continued until the end of rows in the adjacency matrix. In order to minimize the number invalid trees and impossible printing scenarios, following rules are created and will be implemented while generating the matrix:

1) It is important that the generated matrix results in print schedule that has a structure that is layered and not cyclic. A cyclic dependency could result in a scenario where two chunks could have direct or indirect dependencies on each other. For example, in the dependency tree shown in **Figure 4**, currently Node-2 has two dependencies, Chunk-0 and Chunk-1. Allowing cyclic dependency results in the scenario, where Chunk-1 could have dependency on Chunk-2. This can result in stalemate situation where Chunk-2 cannot be printed prior to Chunk-1 and Chunk-1 cannot be printed prior to Chunk-2. In order to check whether the created matrix has cyclic dependency, we take transpose of

the generated matrix and calculate Hadamard product (also known as entry wise product) of the two matrices. If the result of the Hadamard product is not a zero matrix, the generated matrix is ignored as it contains cyclic dependencies between the chunks. Otherwise, the generated matrix is passed on to the next stage.

2) Transitive reduction is used to eliminate double dependency between two chunks. For example, if Chunk-8 has dependent relation with Chunk-1 via Chunk-2, there is no need of edge between Chunk-8 and Chunk-1.

These specified criteria are ingrained into the algorithm that generates random C3DP schedules and thus the generated tree will not violate the rules.

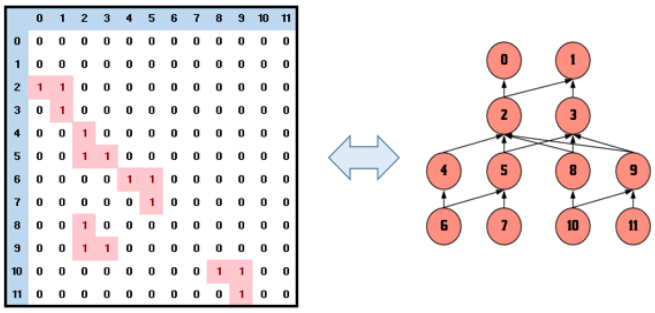

***Figure 4**: Adjacency matrix representing the directed dependency tree*

3.3 Geometric Constraints

The generated DDT regulates both the printing sequence and the dependency relationships between the chunks. However, the generated DDT might not be valid due to potential physical constraints, i.e., the DDT might represent a printing sequence that results in collision between either printing robots (R2R collision) or the printed parts and the printing robots (R2P collision). For example, the print sequence might have two chunks adjacent to each other, being printed simultaneously, which would result in collision between the printing robots. Such scenarios need to be avoided prior to implementing the print schedule. Therefore, a set of constraints must be formulated which a printing strategy can be evaluated against to reject the print strategies that could result in collision, and only accept the collision-free print strategies. These constraints can then serve as a *sufficient* condition for the validation of a printing strategy, i.e., if a printing strategy doesn't violate any of the constraints, the printing strategy is valid and would result in collision-free C3DP. So, any valid printing strategy must satisfy the following:

1) A robot, i, does not collide with already printed chunks. This constraint can be mathematically represented using the concept of accessible space of robot ($AS_{R,i}$), which is 3D space occupied by the printing robot, i, and occupied space of printed chunks (AS_c), which is 3D space that is occupied by the chunks that are already printed. The mathematical formulation of this constraints is presented in Equation (1).

$$AS_{R,i}(t) \cap AS_c(t) = \emptyset, i = 1,2,3 \dots, n \quad (1)$$

2) A robot, i, does not collide with any other robot, j, at any time during the entire printing process. This can be mathematically represented using the concept of swept volume, which is the entire 3D space covered by the printing robot as it prints an assigned chunk. The differentiation between the accessible space and the swept volume of a robot is presented in **Figure 5**. The mathematical formulation of this constraints is presented in equation (2).

$$SV_{R,i}(t) \cap SV_{R,j}(t) = \emptyset, \quad i = 1,2,3 \dots, n;\ j = 1,2,3 \dots, n;\ j \neq i, \quad (2)$$

The discretization of C3DP makes it possible to check the geometric constraints at different time resolution. The check for potential R2R collision is done in the following manner:

- The default check is done at chunk level, i.e., the collision between the two printing robots is checked while they work on their respective chunks. Though this check rejects all the invalid print schedule, it also could potentially reject some valid schedules. This happens when the swept volumes of robots overlap with each other, but the robots might not occupy same exact location at same time. For example, if two robots are working on adjacent chunks, their swept volume will overlap but they might not arrive at the overlapping location at the same time.
- In order to reduce the possibility of false negatives mentioned above (overlapping of swept volume but no collision), the check can be done at layer-level, i.e., the collision between two printing robots is checked while they work on individual level of their respective chunks. Similar to the check at chunk level, this rejects all the invalid, but some valid schedules could be rejected as well.
- To further ensure that false negative has not taken place, we can run check at line-level or G-code level, where a check for collision is done at each G-code line. Though accurate, this is computationally taxing as it requires more frequent checks whereas, the chunk-level check requires least frequent checks.

Thus, first chunk-level check is conducted, if the results are valid, we move on to the next print sequence. Otherwise, layer-level check can be conducted. Similarly, if valid, we can move on to the next layer. Otherwise, line-level check can be conducted. If this produces invalid result, the schedule is discarded as an invalid print schedule. Thus, any generated print strategies can be validated using the geometric constraints presented in Equations (1) and (2) to ensure that the collision does not take place between the printing robots as well as between the printed parts and printing robots.

3.4 Time Evaluation using DDT

In our previous work [7], we presented the idea of using DDT to calculate the total print time of a printing schedule. The number of rows in the trees (i.e., the tree depth) represents the total number of print sequence, i.e., the number of printing steps whereas, the column or the width of the tree represents the number of robots used for parallel printing. For instance, the DDT presented in **Figure 4**, depicts a print schedule with four printing sequence and utilizes maximum of four robots (but only two are needed in two initial sequence). In order to calculate the print time, a recursive function is presented in Equation (3).

$$T(D, c_i) = \max\{[T(D, c_m) | c_m \in c_i.deps], 0\} + t(c_i) \quad (3)$$
$$i = 1,2,3 \ldots . n, \text{ and } m = 1,2,3, \ldots \ldots N_i$$

where, N_i is the number of dependency chunks of c_i. In Equation (3), $T(D, c_i)$ is the time it takes to print chunk, c_i, in a given DDT, D, which is the sum of the time it takes to print the single chunk c_i, i.e., $t(c_i)$ and the time it takes to print all of its dependencies, c_m, i.e., $T(D, c_m)$. This equation can be generalized to calculate the total print time of a DDT as shown in Equation (4), where, T_{total} is the total time needed to print the entire sequence of D, with n chunks. This is equal to the sum of the time it takes to print the last chunk, n, and time it takes to print all of its dependencies, m.

$$T_{total} = \max(\{T(D, c_m) \mid m \in [0, n-1]\}) \quad (4)$$

4. CASE STUDY

To demonstrate how the computational framework works, we present a simple illustrative case study. There are two primary reasons for using a simple model to demonstrate the computational framework: 1) the use of simple model allows audience to focus on the actual computational framework rather than getting distracted by the complexity of the model (and its chunking) and jargons associated with the additional complexity. 2) The use of simple model allows audience to visualize the sloped-interface chunking strategy in an intuitive manner. The chunking of simple geometry results in regular geometric shapes which makes it much easier to understand and visualize how the geometric constraints translate to actual physical constraint.

The part considered for this case study is a rectangular block for demonstration purpose. The dimensions of block are $100\text{cm} \times 80\text{cm} \times 1.5\text{cm}$ and has a total volume of $12{,}000\ cm^3$. The block is printed using PLA. The rectangular block and the resulting chunks (chunked using sloped-interface chunking method) obtained after chunking is presented in **Figure 5**. Four robots are used for this case study. In addition, the following assumptions are made:

1) The chunks created have equal volume and can only have one of the shapes shown in **Figure 5**. If different chunking strategy is chosen for chunking, the shape of the chunks could be different. Since the volume is equal, and the printing parameters are the same for all the printers, the time to print each chunk is assumed to be equal for simplifying the evaluation of print time. Assuming the material is deposited at the rate of 16 $mm^3/_s$ using a 0.4 mm nozzle, the estimated time to print a chunk is 10.42 hours.
2) Once a chunk is printed, printing robot moves to the location of next chunk immediately. The travel time between the chunks is ignored in the calculation.

The output of chunker (i.e., the chunking algorithm) consists of eight coordinate points, four coordinates for four corners of the base and the other four coordinates for the top corners of the chunks. In addition to this, chunk number is also outputted. The eight coordinates are used for checking constraints and the chunk number is used for generating adjacency matrix as well as a DDT. The following steps were taken to implement the computational framework in this case study:

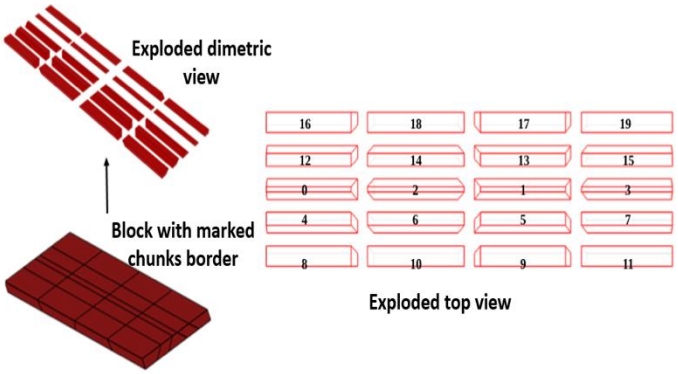

***Figure 5**: Rectangular block showing the chunks line, exploded view of the chunks that combine to make a rectangular block and top view of exploded chunks with chunk number marked*

1) Generation of print schedule for rectangular block

The result of chunking is shown in **Figure 5**. The algorithm then generates adjacency matrix that meets all the predefined criteria specified in **Section 3.2**. This generated matrix represents a print schedule. The next step is to check the validity of the print schedule using the geometric constraints presented in **Section 3.2**.

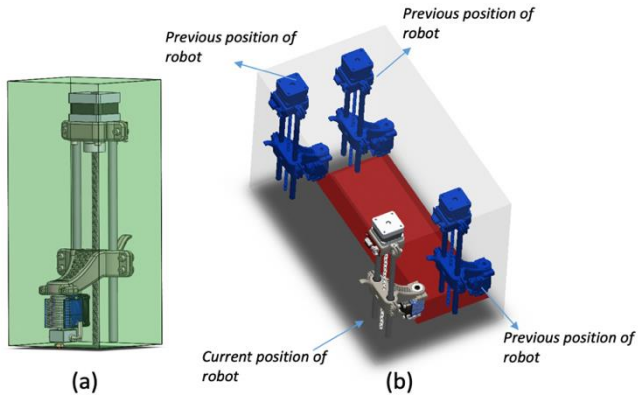

***Figure 6**:(a) Accessible Space of robot (reduced to z-stage) (b) Swept Volume of robot while print a chunk. Position of robot shown at different chunk corner coordinate*

2) Validation check of generated schedules using geometric constraints

In order to check the geometric constraints, the swept volumes (SV) of the active robots are defined as shown in the

Figure 6. The algorithm checks for overlap between the swept volumes of the printing robots (for R2R collision check). The second type of check is conducted between the printing robot and the already printed part (R2P collision check). First, the accessible space (AS_R) of the robot is defined. In this case study, a reduced constraint was used to define AS_R. This constraint is generated by considering only the z-stage of the print robot, shown in

Figure 6(a)[2]. After that, the occupied space (AS_C) by the chunk is defined using the eight coordinates outputted by the chunker. The AS_C for each individual chunk is defined using a list of its individual corner coordinates.

The algorithm goes through the sequence and does all the constraint checks. For example, if there are multiple chunks being printed in a sequence, it checks for R2R collision using swept volume of the involved robots. If there is no collision, the chunks coordinates are stored in a printed chunk list so that they can be used for R2P collision check during subsequent sequences. If the print schedule does not violate either of the geometric constraints, the DDT is considered valid, otherwise discarded as invalid.

3) *Time evaluation of the valid print schedules*

For this case study, 1000 DDTs were randomly generated first, and the computational framework returns us with 60 valid trees. Rest of the 940 trees were either invalid or duplicates of valid trees. Upon the completion of generation and constraints check, the valid print schedules were evaluated using time metrics presented in **Section 3.4**. Each chunk takes about 10.42 $hours$ to print, which is the maximum time it takes to complete each sequence.

5. RESULTS AND DISCUSSION

Once the valid trees were evaluated, all of the 60 trees were ranked based on the total time it takes to print the entire print sequence. The validity of the generated valid trees is also double-checked by hand to ensure that the algorithm works as intended. The top five print time generated using the algorithm along with the one created using heuristic approach are presented in the **Table 1**.

Table 1.0 *Top five print schedule generated using the algorithm in addition to the one generated using the heuristic approach (labelled "H") and their total print time*

Label	Print Sequence	Time (hrs)
1	{0,0,1,1,2,3,4,5,6,7,8,9,2,3,6,6,8,9,10,10}	114.62
2	{0,0,1,1,2,3,4,4,5,6,7,8,3,2,4,5,6,7,8,9}	104.2
3	{0,0,1,1,2,2,3,4,6,5,7,8,3,2,4,5,6,7,8,9}	104.2
4	{0,0,1,1,2,2,5,3,6,6,8,7,3,4,5,6,6,7,8,9}	104.2
5	{0,0,1,1,2,2,4,3,7,6,8,7,2,3,5,6,4,7,8,9}	104.2
H	{0,0,1,1,2,2,3,3,4,4,5,5, 2,2,3,3,4,4,5,5}	62.52

The print sequence in Table 1 is presented in list format, where each element represents the sequence number and *not* the chunk number. The first element of the list represents the sequence for Chunk-0, second element represents the sequence of Chunk-1 and third element represents the sequence number of Chunk 3 and so on, giving us total of 20 elements. For example, for print sequence labelled 1, first two elements are both 0, which means Chunk-0 and Chunk-1 are printed together during the first print sequence (labelled 0). The third and fourth elements are both 1, which means Chunk-2 and Chunk-3 are printed during the second print sequence (labelled 1). The fifth element is 2, which means Chunk-4 is printed during the third print sequence (labelled 2). This process goes on until the end of the list.

The top five generated print schedule are presented in DDT format in **Figure 8** and the print sequence developed using heuristic approach is presented in **Figure 7**

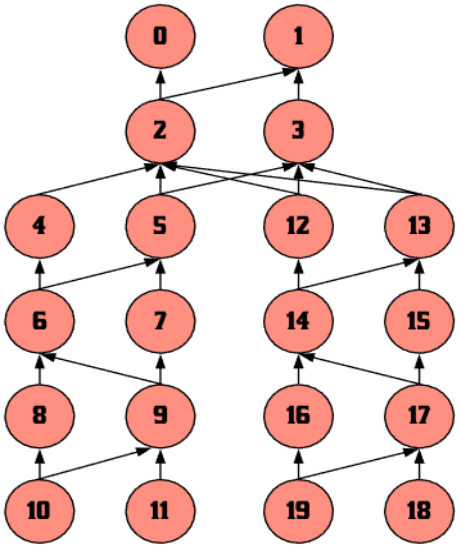

Figure 7: *The dependency tree associated with the print sequence developed using heuristic approach*

[2] The assumption is made in order to make it more applicable to robotic arm/ Scara arm 3D printers.

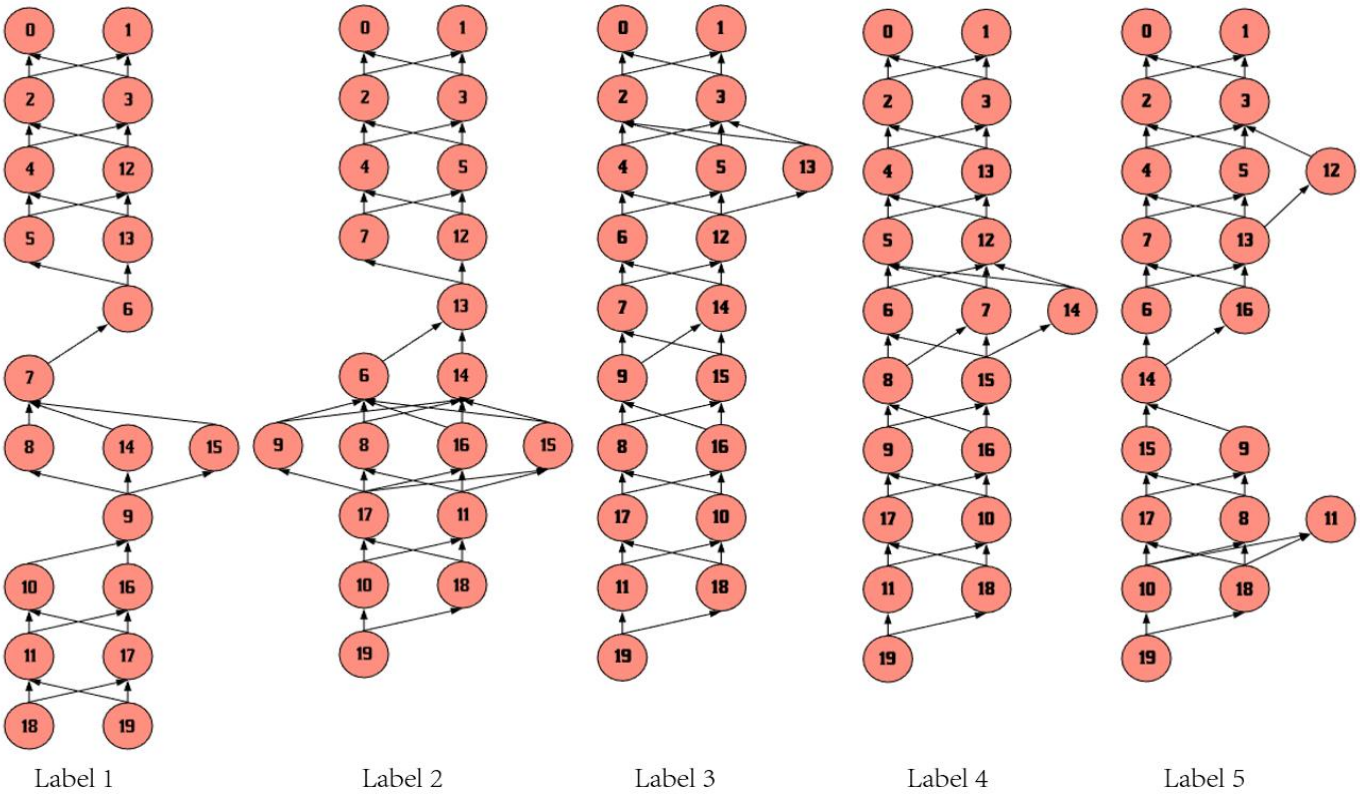

***Figure 8**: The dependency tree associated with the print sequence represented in Table 1.0*

The total print time for the print schedules generated using the algorithm is much longer (~1.67 times longer) compared to the one for heuristically created. This is clear when we take a look at the tree presented in **Figure 7** and **Figure 8**. While the heuristically generated print schedule utilizes most of the available printing robots (two while printing the initial chunks in center row and four afterwards), the automatically generated only utilizes two or three at a time leaving spare printers without use. In subsequent version of the algorithm development, we could introduce some sort of incentive in the algorithm for utilizing more robots to shorten the print time. Though the total print time of heuristic approach was faster for this case, we'd like to highlight that this study is the stepping stone towards the bigger picture – to develop a computational framework that can take the input from chunking algorithm to automatically generate print schedules for objects with very complex geometries. In those situations, human heuristics would easily fail due to the cognitive limitations of human brain. In this study, we present a case study with simple geometry for the illustrative purpose of demonstrating the entire process and the framework.

In order to see how the distribution of total print time looks like for the generated valid print schedules, we plotted the total print time of every generated print schedules against the label number of the generated valid schedules. The plot is presented in **Figure 9**. In order to compare the total print time between the print schedules generated using the algorithm with the heuristic approach, a reference line (i.e., the print time form the heuristic approach) is plotted in the graph**.**

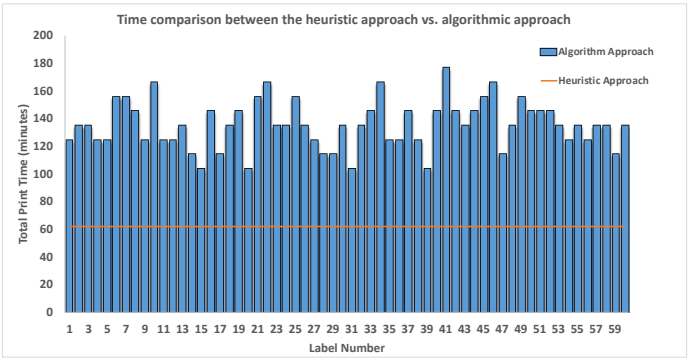

***Figure 9**: The plot showing the total print time for each valid generated tree and the total print time for schedule generated using heuristic approach*

6. CONCLUSION

In this paper, a printing schedule of cooperative 3D printing (C3DP) for a given number of chunks was algorithmically created with specified number of robots. The generated schedule is validated using the newly developed geometric constraints for cooperative 3D printing. These geometric constraints check for collision between the robots (R2R) while they are working in parallel as well as for collision between the printing robots and

the printed parts (R2P). If a generated schedule does not satisfy the geometric constraints, they are rejected as invalid. The list of valid trees is then evaluated in order to calculate the total print time for each tree, using the time metrics developed. The valid print schedule with shortest print time is chosen for printing. The key contributions of this study are:

- Development of print sequence generator that can automatically generate a print schedule for specified number of chunks and available number of robots using output of chunker.
- Identification and mathematical formulation of geometric constraints that can check the validity of the generated print schedules.
- Development of evaluation time metric using directed dependency tree (DDT) to determine the total print time for each valid print schedule.
- Development of computational framework that amalgamates print sequence generator, geometric constraint check and time evaluation metric that can automatically generate, validate and, evaluate print schedule for given chunking strategy.

While we successfully developed algorithm to generate print schedule for given number of chunks, the generated schedules take much longer to complete printing compared to the heuristic approach. The logical step is to integrate an optimization framework to obtain collision-free print schedule that take shorter make span than the heuristics. In addition to this, all the robots were not utilized while printing so, we can implement incentives to promote higher utilization of available robots in subsequent version of the algorithm.

ACKNOWLEDGEMENT

The authors would like to thank Jace McPherson for his contribution in helping us create multi-robot print simulations for C3DP.

REFERENCES

[1] L. J. Love, “Utility of Big Area Additive Manufacturing (BAAM) For The Rapid Manufacture of Customized Electric Vehicles.” United States. Dept. of Energy. Office of Energy Efficiency and Renewable Energy ;, Washington, D.C. :, 2015.

[2] Autodesk, *Project Escher*. 2016.

[3] Y. Jin, H. Pierson, and H. Liao, *Toolpath Allocation and Scheduling for Concurrent Fused Filament Fabrication with Multiple Extruders*. 2017.

[4] L. G. Marques, R. A. Williams, and W. Zhou, “A Mobile 3D Printer for Cooperative 3D Printing.”

[5] J. McPherson and W. Zhou, “A Chunk-based Slicer for Cooperative 3D Printing,” *From Rapid Prototyp. J.*, vol. Accepted, 2018.

[6] L. Poudel, Z. Sha, and W. Zhou, “Mechanical Strength of Chunk-Based Printed Parts For Cooperative 3D Printing,” in *46th SME North American Manufacturing Research Conference, NAMRC 46*, 2018, vol. Accepted.

[7] L. Poudel, C. Bair, J. McPherson, Z. Sha, and W. Zhou, “A Heuristic Based Scaling Strategy For Coperative 3D Printing,” *Rapid Prototyp. J.*

[8] T. Kis, “Job-shop scheduling with processing alternatives,” *Eur. J. Oper. Res.*, vol. 151, no. 2, pp. 307–332, Dec. 2003.

[9] M. Petrović, N. Vuković, M. Mitić, and Z. Miljković, “Integration of process planning and scheduling using chaotic particle swarm optimization algorithm,” *Expert Syst. Appl.*, vol. 64, pp. 569–588, Dec. 2016.

[10] G. Chryssolouris, S. Chan, and W. Cobb, “Decision making on the factory floor: An integrated approach to process planning and scheduling,” *Robot. Comput. Integr. Manuf.*, vol. 1, no. 3–4, pp. 315–319, Jan. 1984.

[11] R. M. Sundaram and S.-S. Fu, “Process planning and scheduling\—a method of integration for productivity improvement,” *Comput. Ind. Eng.*, vol. 15, no. 1, pp. 296–301, 1988.

[12] N. MORAD and A. M. S. ZALZALA, “Genetic algorithms in integrated process planning and scheduling,” *J. Intell. Manuf.*, vol. 10, no. 2, pp. 169–179, Apr. 1999.

[13] X. Shao, X. Li, L. Gao, and C. Zhang, “Integration of process planning and scheduling—A modified genetic algorithm-based approach,” *Comput. Oper. Res.*, vol. 36, no. 6, pp. 2082–2096, Jun. 2009.

[14] X. Li, L. Gao, X. Shao, C. Zhang, and C. Wang, “Mathematical modeling and evolutionary algorithm-based approach for integrated process planning and scheduling,” *Comput. Oper. Res.*, vol. 37, no. 4, pp. 656–667, Apr. 2010.

[15] C. W. Leung, T. N. Wong, K. L. Mak, and R. Y. K. Fung, “Integrated process planning and scheduling by an agent-based ant colony optimization,” *Comput. Ind. Eng.*, vol. 59, no. 1, pp. 166–180, Aug. 2010.

[16] M. Koes, I. Nourbakhsh, and K. Sycara, “Heterogeneous Multirobot Coordination with Spatial and Temporal Constraints,” in *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3*, 2005, pp. 1292–1297.

[17] N. Mishra, B. B. Choudhury, and B. B. Biswal, “An effective technique for generation of assembly sequence in multi-robotic assembly cells,” in *2012 NATIONAL CONFERENCE ON COMPUTING AND COMMUNICATION SYSTEMS*, 2012, pp. 1–5.

[18] J. Hun Park and M. Jin Chung, “Automatic generation of assembly sequences for multi-robot workcell,” *Robot. Comput. Integr. Manuf.*, vol. 10, no. 5, pp. 355–363, Oct. 1993.

[19] M. Gombolay, R. Wilcox, and J. Shah, “Fast scheduling of multi-robot teams with temporospatial constraints,” 2013.

[20] W. Wan, K. Harada, and K. Nagata, “Assembly sequence planning for motion planning,” *Assem. Autom.*, vol. 38, no. 2, pp. 195–206, Apr. 2018.