A Generative Approach for Scheduling Multi-Robot Cooperative 3D Printing

Laxmi Poudel
Graduate Research Assistant, SiDi Lab
University of Arkansas, Fayetteville AR 72701
lpoudel@uark.edu
ASME student member

Wenchao Zhou
Assistant Professor, AM^3 Lab
University of Arkansas, Fayetteville AR 72701
zhouw@uark.edu
ASME member

Zhenghui Sha[1]
Assistant Professor, SiDi Lab
University of Arkansas, Fayetteville AR 72701
zsha@uark.edu
ASME member

ABSTRACT

Cooperative 3D printing (C3DP) is a novel approach to additive manufacturing, where multiple printhead-carrying mobile robots work cooperatively to print the 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 robots to print and bond. These robots will work simultaneously in a scheduled sequence to print the entire part. Although promising, C3DP lacks a generative approach that enables automatic chunking and scheduling. In this study, we aim to develop a generative approach that can automatically generate different print schedules for a chunked object by exploring a larger solution space that is often beyond the capability of human cognition. The generative approach contains 1) a random generator of diverse print schedules based on an adjacency matrix that represents a directed dependency tree 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 evaluator for comparing the performance of all valid schedules. We demonstrate the efficacy of the generative approach using two case studies: a large simple rectangular bar and a miniature folding SUV with more complicated geometry. This study demonstrates that the generative approach can generate a large number of different print schedules for collision-free C3DP which cannot be explored solely using human heuristics. This generative approach lays the foundation of building the optimization approach of C3DP scheduling.

1. INTRODUCTION

As additive manufacturing (AM) is transitioning from rapid prototyping to digital manufacturing over the past years, the current cost structure of the technologies is a major barrier for mainstream manufacturing adoption. One critical factor of the cost structure is the scalability, in terms of both print size and printing speed. Extensive

[1] Corresponding author: zsha@uark.edu

research has been performed on increasing the size of the printer for printing larger parts, e.g., BAAM (Big Area Additive Manufacturing) developed by Oak Ridge National Lab [1] and Sciacky EBAM 300. Also, there is other research 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 as 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 an individual layer of the 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 a 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.

While some of the aforementioned approaches have 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 a larger part but the print quality will lower as a result of the coarser resolution. To achieve a balance for this “incongruous 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 works simultaneously to print and assemble large objects [4]. One of the most important features of SPA, which distinguishes itself from the traditional layer-based 3D printing system is the chunk-based printing strategy – a 3D model of the desired part is divided into smaller chunks first and each of these chunks is assigned to individual printing robots. Each robot prints one chunk at a time, but many printing robots work in parallel, and layer by layer for each chunk as illustrated in

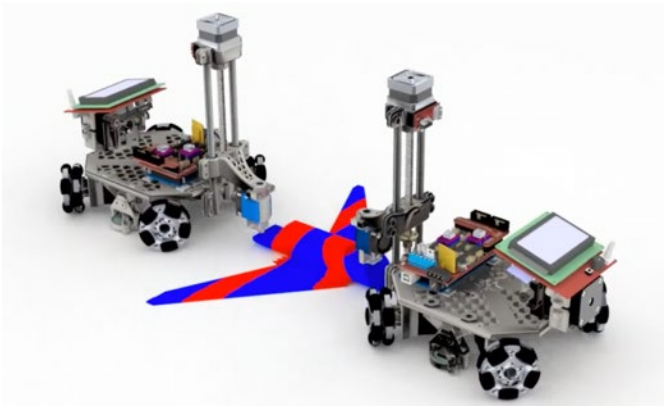

***Figure 1**: Chunk-based Cooperative 3D printing*

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.

In our preliminary work, we have developed a chunk-based slicer 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][7], which shows that the chunk-based 3D printed part has a comparable tensile strength to traditional layer-based 3D printed part. To scale C3DP to multiple robots in [8], 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. These robots then move over to print the remaining chunks in the same row 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 the second row of chunks on the opposite side of the central row as shown in **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. To implement the process of C3DP, the software, as well as the hardware platform integration along with the entire system architecture, is demonstrated in [9].

While a print schedule based on this simple heuristic with a finite number of robots might give us good results, as the number of chunks increases, the number of possible print schedules scales with *n!* where *n* is the number of chunks, and therefore it becomes challenging for humans to explore the large solution space solely based on heuristics. SPAR3 was the only valid scheduling strategy we were able to identify based on human heuristics in our previous work [8]. Therefore, the large portion of the solution space of possible printing strategies remains unexplored. Hence, we cannot verify whether the optimal print schedule has been achieved indeed. Thus, we must search the entire

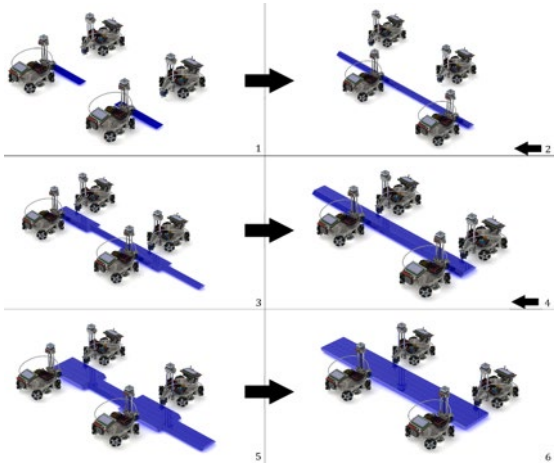

***Figure 2**: SPAR3 strategy illustration*

solution space to explore other viable scheduling strategies, which is crucial to the development of optimization of C3DP scheduling. The challenge lies in being able to generate diverse valid print schedules in the presence of complex geometric and temporal constraints in C3DP. In this paper, we present a new generative approach that can automatically generate diverse printing schedules for a given number of chunks and robots and evaluate the validity of the generated schedules against the constraints in space and time.

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 generative approach, which includes the model for C3DP, geometric constraints, and time evaluation metrics. Case studies are presented in Section 4 that demonstrate the utility and performance of the proposed approach. Finally, the results of the case studies and the discussion are presented in Section 5, followed by the 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 maximize parallel printing and minimizes the total time of printing,) 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 the IPPS problem further, making it an 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 metaheuristics approach (e.g., Genetic Algorithm [11-14]) and agent-based approaches (e.g., Particle Swarm Optimization [11] and Ant Colony Optimization [17]), the problem definition is limited around the job shop scheduling problems, where multiple jobs need to be scheduled in multiple workstations. Scheduling plans for robotic applications is not studied in most of the related research. The 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 the IPPS problem[11]. Though the work presents and demonstrates the motion trajectories of mobile robots, the scope of work does not include collision detection between mobile robots, where spatiotemporal constraints need to be developed and applied.

Similarly, multi-robot systems (MRS) is another related 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 problems as mixed-integer linear programming (MILP) problem, which could outperform greedy heuristics, and market-based approaches which

separates scheduling and task allocation [18]. In a related study, a connectivity graph along with the Liaison method was used to generate a sequence for multi-robot assembly by Mishra et al. [19]. Parallel execution of assembly sequences for multi-robot work cells was studied by Park et al. [20]. While they developed constraints to avoid infeasible assembly sequence, no constraints were developed to avoid a robot-to-robot collision as the only single gripping robot was used for demonstration. A novel algorithm, Terico, was developed to generate schedule fulfilling temporospatial constraints for the multi-robot system by Gombolay et al. [21]. 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 behaves as the number of robots is greater than 10. Wan et al. proposed a newly developed planner for finding optimal assembly sequences to assemble objects and demonstrated the results by optimally scheduling Soma cube [22]. However, this work is only applicable to single robots, which eliminates the complexity associated with the collision between working robots. More recently, Shriyam et al. proposed a model that allows multiple mobile teams of robots to execute specified tasks. These tasks can be interrupted and robots that are assigned to the tasks can be rescheduled from an unfinished task to a different task. They compared five different task assignment and scheduling algorithms that use different task prioritization heuristic and compared their results. They concluded that the strategy that ranks the tasks based on the execution duration using the concept of static levels provided the best outcome [23]. Bhatt et al. demonstrated the use of multiple manipulators with 3 degree of freedom to print free-form thin shell parts using support 3D printing [24]. Though the developed approach is capable of printing complex structures, the manipulators are fixed and possess less complexity than the problem in C3DP but could potentially be integrated into C3DP platform in the future for printing complex structures with thin shell parts without support structures.

All of the literature discussed above, both in the field of IPPS and MRS, and other extant literature provides different solutions to multi-robot planning and scheduling problem. Although 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 a 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 need to be completed. The problem of C3DP, on the other hand, is a unique blend that encompasses both 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. Also, the integration of 3D printing makes the manufacturing process continuous, where the material is continuously deposited temporospatially until the desired part is manufactured. There is no existing method that could take the design of the desired part and implement 3D printing using multiple printing robots as highlighted by Bhatt et al. in [25], where the author highlights the need of multi-robot systems in additive manufacturing to reduce the total print time using conformal and multi-resolution

printing. Thus, there is a clear gap for generating viable and valid chunking and scheduling strategies for C3DP. This paper aims to create a generative approach that allows the automatic generation of valid scheduling strategies with the given number of chunks and robots by considering and establishing the temporospatial constraints that are unique to C3DP.

3. RESEARCH APPROACH

3.1 The manufacturing stages in C3DP

Before jumping into the details of the generative approach, 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 the multi-stage discrete process as illustrated in **Figure 3** such that there are inter-dependencies between multiple stages.

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 3** [8]. A DDT defines 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 generative approach, a random DDT is first generated, which contains the scheduling information such as chunk dependencies and the number of sequences

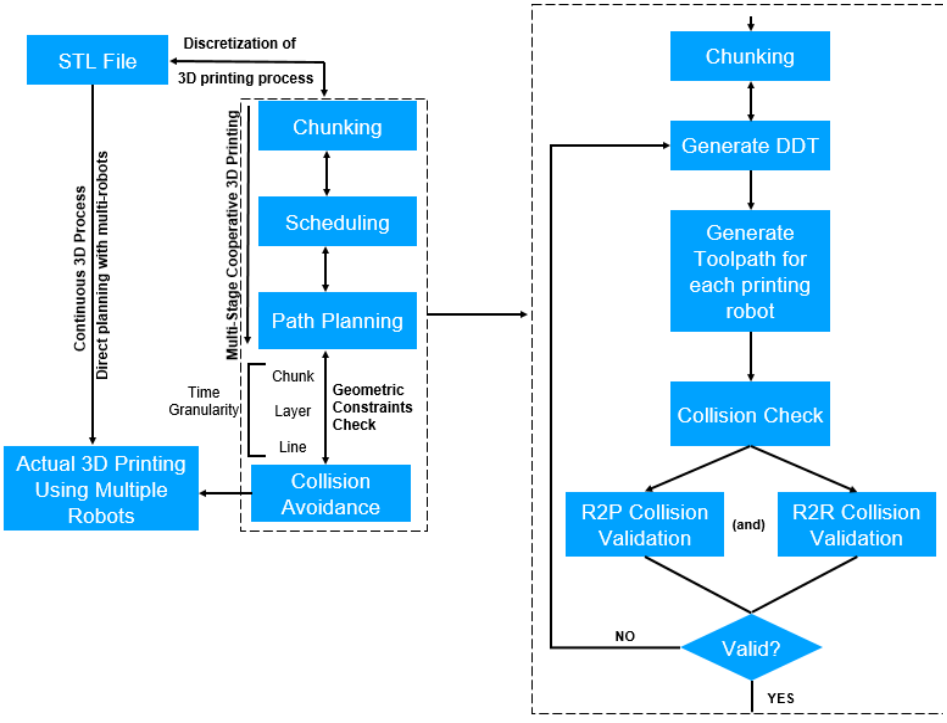

***Figure 3**: The different stages of C3DP*

(i.e., the depth of a tree). To generate a DDT, two different types of information are needed: information related to the geometric dependencies between the chunks, which is generated during the chunking process, and the information related to the ordering of the chunk for printing. While the geometric dependency relationship is vital it is not enough to generate a DDT because it provides no information about the relationship between the chunks that do not have geometric dependency between them. For example, in **Figure 7**, what is the relationship between chunk 6 and chunk 17? And how are they placed in a DDT? Human heuristics were used to concatenate this information with the geometric dependencies to generate a full DDT in our previous studies [8]. Due to the limitation of human cognition, human heuristics might not be able to explore every possibility and there could be many more DDTs that are also valid and maybe even better than the one generated by a human. Therefore, rather than using human heuristics to do so, we automatically generate many DDTs using the automatic generator. In doing so, this process eliminates any human bias based on prior knowledge, out of the generative process, and explores the space of which human heuristics might not be capable.

Then, the path planning is done for the generated DDT. Before 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 generative approach of automatically scheduling are depicted in **Figure 3**.

3.2 Random Generation of Print Sequence

For the random generation of C3DP schedules, a part is first divided into n chunks by a chunker. We then use $G(n, p)$ method (The Erdos-Rényi method) [22,23] to generate a random graph. In this approach, once the chunking is complete, an adjacency matrix of dimension $n \times n$ is generated and initialized as zero matrix. After that 0's are replaced with 1's randomly using a random function. The value of 1 represents a dependent relationship between two chunks and 0 represents the 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 represent the same information. 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 are continued until the end of rows in the adjacency matrix. To minimize the number of invalid trees and impossible printing scenarios, the following rules are created and will be implemented while generating the matrix:

1. The generated matrix must result in a 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 a dependency on Chunk-2. This can result in a stalemate

situation where Chunk-2 cannot be printed before Chunk-1 and Chunk-1 cannot be printed before Chunk-2. To check whether the created matrix has a cyclic dependency, we take the 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[28] 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 for the 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.

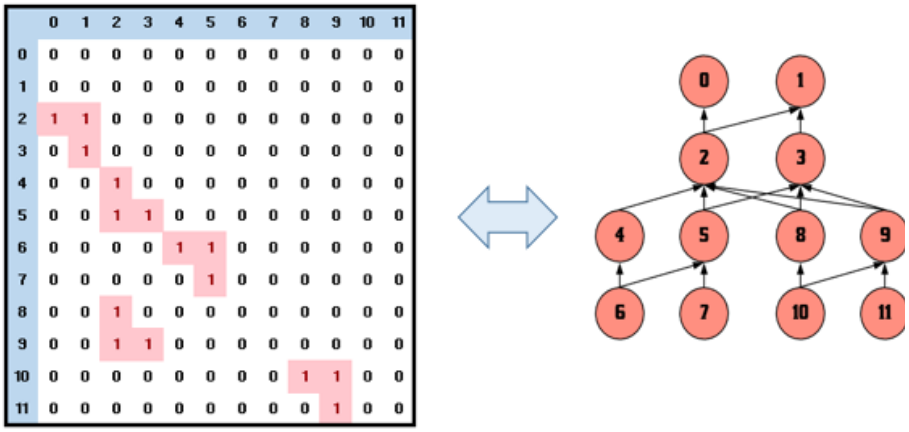

***Figure 4**: Adjacency matrix representing the directed dependency tree*

3.3 Geometric Constraints

In our previous studies, we identified geometric constraints to check the validity of the print schedules to ensure that the generated print schedule results in collision-free printing [7]. The identified geometric constraints check for two types of collision: the collision between the active printing robots (R2R collision) and collision between the active robot and the previously printed part (R2P collision). Following geometric constraints were identified in our previous studies [7]:

1) A robot, i does not collide with already printed chunks. The mathematical formulation of these constraints was presented in the form of Equation (1) using the concept of accessible space of the robot (smallest cuboid enclosing the robots) and occupied space of the printed chunk (3D shape of a printed chunk).

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

Where, $AS_{R,i}(t)$ *is the accessible space of the robot*, i *at time*, t

$AS_c(t)$ *is the occupied space of printed chunk*, c *at time*, t

2) A robot, i, does not collide with any other robot, j, at any time during the entire printing process. The mathematical formulation of these constraints was presented in the form of equation (2) using the concept of swept volume of the printing robot (3D space occupied by the robot as it prints the entire chunk).

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

Where, $SV_{R,i}(t)$ *and* $SV_{R,j}(t)$ *are the swept volume of the robot i and robot j at time t*

The detailed definitions of the accessible space occupied space, and swept volume, as well as the detailed description of the identified geometric constraints, can be found in our previous studies [7]. 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 [8], 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 sequences, 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 a maximum of four robots (but only two are needed in two initial sequences. In the study, we presented Equation (3) to calculate the total print time of a DDT, 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, c_n, time it takes to print all of its dependencies, c_m.

$$T_{total} = \max(\{T(D, c_m) \mid m \in [0, n-1]\}) \quad (3)$$

4. CASE STUDIES

To demonstrate how the generative approach works, we present two illustrative case studies. The first case study is a large-scale rectangular bar with simple geometry. There are two primary reasons for adopting a simple model in the first place: 1) it allows us to better demonstrate the generative approach for generating different schedules. 2) It allows us to visualize the sloped-interface chunking strategy intuitively. The chunking of simple geometry results in regular geometric shapes which better visualizes how the geometric constraints translate to actual physical constraints. The second case study includes a miniature folding SUV vehicle with more complex geometries and irregular chunk shapes, which was chosen for demonstrating the generalizability and versatility of the generative approach. In the case studies, we have ignored the traveling time between chunks, i.e., once a chunk is printed, the printing robot moves to the location of the next

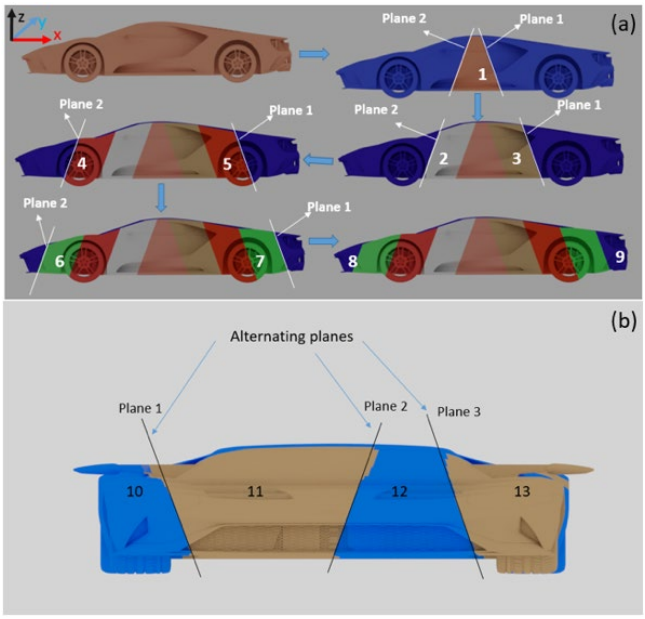

***Figure 5**: (a) Chunking done along X-direction, where chunks are created by using cutting plane 1 and plane 2 (b) Chunking done along Y-direction, where chunks are created by alternating planes along the direction.*

chunk immediately. Ignoring the travel time between the chunks would not result in significant difference due to a) out of total time, majority of the time is spent on printing large chunks compared to traveling between the chunks (roughly more than 95%), and b) the travel speed is usually much faster (more than 2X) than the print speed. Thus, the travel time can be ignored without resulting in a significant discrepancy in calculating the total print time.

4.1 Summary of sloped surface chunking strategy

To generate chunks, we adopted a sloped surface chunking strategy developed previously [6] for the FDM process. It allows the chunks to be bonded together during the 3D printing process without post-processing and assembly. In the sloped surface chunking, a center chunk is created along one axis (e.g., in the X direction) using two cutting planes (plane 1 and plane 2) as shown in **Figure 5(a)**. Once the center chunk is created, these planes are shifted outward along the axis as shown in the figure. The chunks are generated till the end of the part is reached. Once the chunking is completed along the axis, chunking takes place along another direction (e.g., in the Y direction), which follows the same approach. The only difference is the direction of the surface normal of the cutting planes at every location as shown in **Figure 5(b)**, where plane 1 and plane 3 are parallel whereas plane 2 in-between has the opposite direction. This is done so that multiple chunks can be printed simultaneously.

This chunking approach allows the chunks to be bonded during the printing process, similar to how the layers are bonded together in the FDM process. **Figure 6(a)** depicts how the chunk bond is formed while the chunks are being printed. Once chunk 1 is printed, the printer starts working on chunk 2, which shares the sloped surface with chunk 1. As the heated nozzle starts printing chunk 2, it comes in contact with the surface of

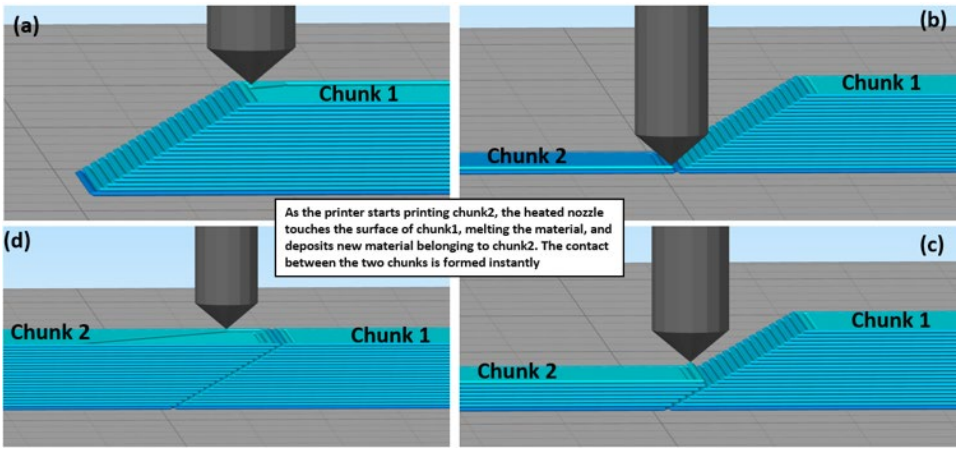

Figure 6: *The process depicting how the connection between the chunks is formed. (a) Chunk 1 is printed (b) beginning of Chunk 2, adjacent to chunk 1, being printed (c) New molten material being deposited on sloped surface of chunk 1 belonging to chunk 2 (d) Full chunk bond is formed as chunk 2 printing is almost complete*

chunk1 (**Figure 6(b)**) and melts the surface of the chunk. As the sloped surface of chunk 1 is being melted by the hot nozzle, the new material is deposited as well, which helps form contact between the two hot molten surfaces and result in instant adhesion between the chunks. We have studied the mechanical strength of the part created using this chunking strategy [6], and the results show that the chunk-printed part can be made as strong as the standard 3D printed part when the chunking parameters are appropriately selected.

4.2 **Case Study I: Rectangular bar**

The part considered for this case study is a rectangular block for demonstration purposes. The dimensions of the block are $100cm\ X\ 80cm\ X\ 1.5cm$ and have 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 7**. Four robots are used for this case study. In addition, the following assumption is made: The chunks created have equal volume and can only have one of the shapes shown in **Figure 7**. 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 $0.4mm$ nozzle, the estimated time to print a chunk is 10.42 hours.

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, the 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 generative approach in this case study:

1. *Generation of print schedule for a rectangular block*

The result of chunking is shown in **Figure 7**. The algorithm then generates an adjacency matrix that meets all the predefined criteria specified in Section 3.2. This

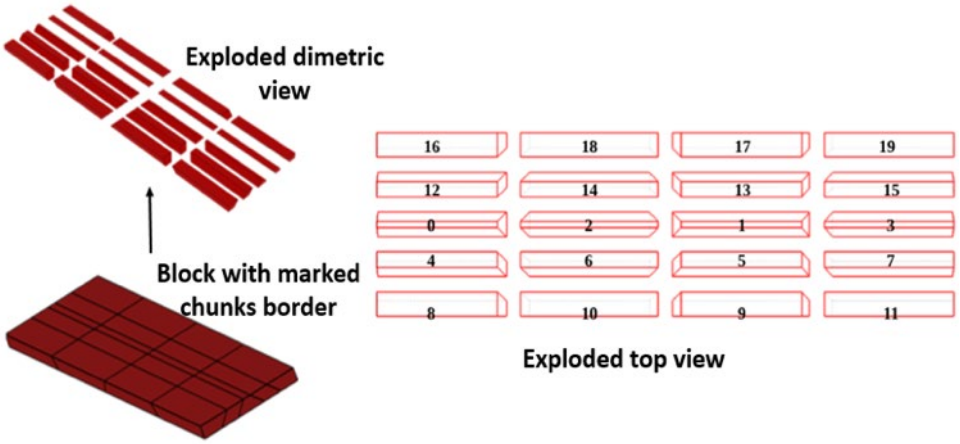

Figure 7: *The rectangular block showing the chunks line, the exploded view of the chunks and the top view of exploded chunks with chunk number marked*

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.

2. *Validation check of generated schedules using geometric constraints*

To check the geometric constraints, the swept volumes (SV) of the active robots are defined as shown in **Figure 8**. 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 8 (a)**[2]. After that, the occupied space (AS_C) by

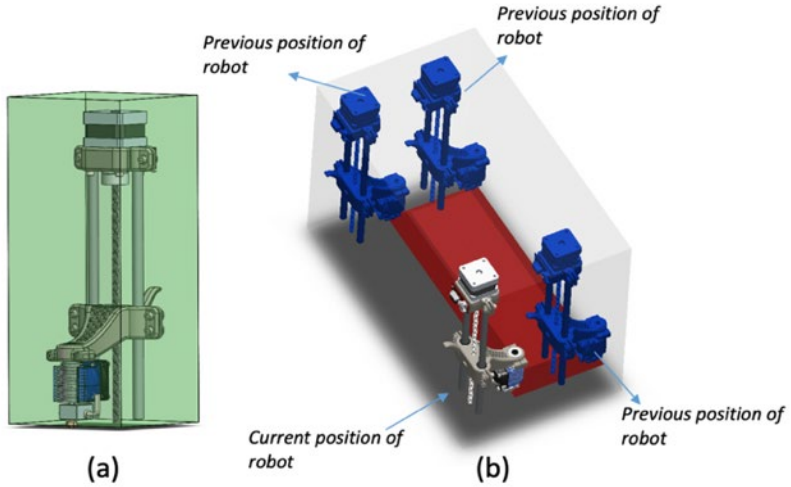

Figure 8 *:(a) Accessible Space of the robot (reduced to z-stage) (b) Swept Volume of the robot while print a chunk. Positions of the robots shown at different chunk corner coordinates*

[2]The assumption is made in order to make it more applicable to robotic arm or scara arm 3D printers. At the time, while this paper was being written, the transition was being made from a previous generation mobile robot with printhead as show in Figure 1 to more robust mobile robot with scara arm that has longer reach.

the chunk is defined using the eight coordinates outputted by the chunker. The AS_C for each chunk is defined using a list of its corner coordinates.

The algorithm goes through the sequence and does all the constraint checks. For example, if multiple chunks are being printed in a sequence, it checks for R2R collision using the swept volume of the involved robots. If there is no collision, the coordinates of the chunk are stored in a printed chunk list so that they can be used for the 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 generative approach returns us with 60 valid trees. The 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.

4.3 **Case Study II: Folding SUV vehicle**

For the second case study, we use a toy folding SUV with a dimension of $157.6\ cm \times 140\ cm \times 3.6\ cm$. The STL model for the folding SUV vehicle was obtained from Fab365 – an e-product marketplace for 3D printing. The print object used for case study II has

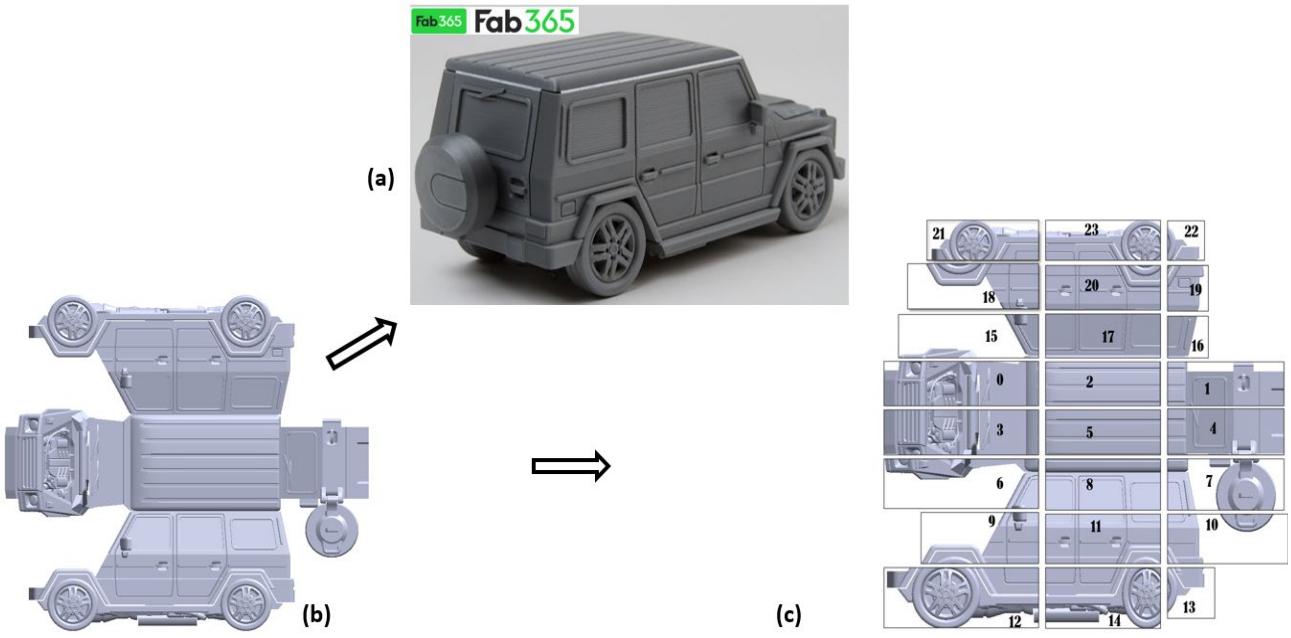

Figure 9*: (a) Folded 3D model of a printed SUV vehicle (b) Top view of the unfolded STL model of the SUV vehicle (c) Top view of the exploded chunks*

With this new development, the check only needs to be conducted to check for collision between the scara arms of working robots rather than the entire body of the robot.

more complex geometry and as a result of this, the chunking results in chunks with variable sizes, volumes, and shapes. The part and the resulting chunks obtained after chunking is presented in **Figure 9**. Since the part is larger than the first case study, to reduce the total print time, $40\ mm^3/s$ deposition rate was used for time calculation.

Similar to the first case study, the chunker output eight coordinates for each chunk. To reduce the computation, we use the minimum enclosing volume (MEV) to represent a chunk, defined as the smallest trapezoidal prism (for center row chunks) or parallelepiped (non-center row chunks) that would enclose the entire chunk just like the one shown in **Figure 10**. This is a conservative approach and might eliminate some valid print schedules during collision check, but it drastically reduces computational resources required to store geometrical information and check geometric constraints.

The print schedule for twenty-four chunks (numbered $0-23$) of folding SUV is generated using the same approach outlined in case study I. But unlike the first case study, the chunks generated did not have uniform volume. Once the generation of print schedules is completed, these generated schedules go through constraints validation to ensure they result in collision-free printing. We use the MEV shown in **Figure 10(b**) to define the occupied space of the chunk ($AS_c(t)$), accessible space of the robots ($AS_{R,i}(t)$), and the swept volume of the robots ($SV_{R,i}(t)$) to check for collision during printing. The invalid print schedules are rejected, and the valid ones are passed on to the time evaluation stage of the generative approach. In the final stage, we calculate the total print time of all the valid print schedules using the actual volume of the chunk instead of the approximated volume to get a more accurate result.

5. RESULTS AND DISCUSSION

5.1 Case Study I

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 a heuristic approach are presented in **Table 1**.

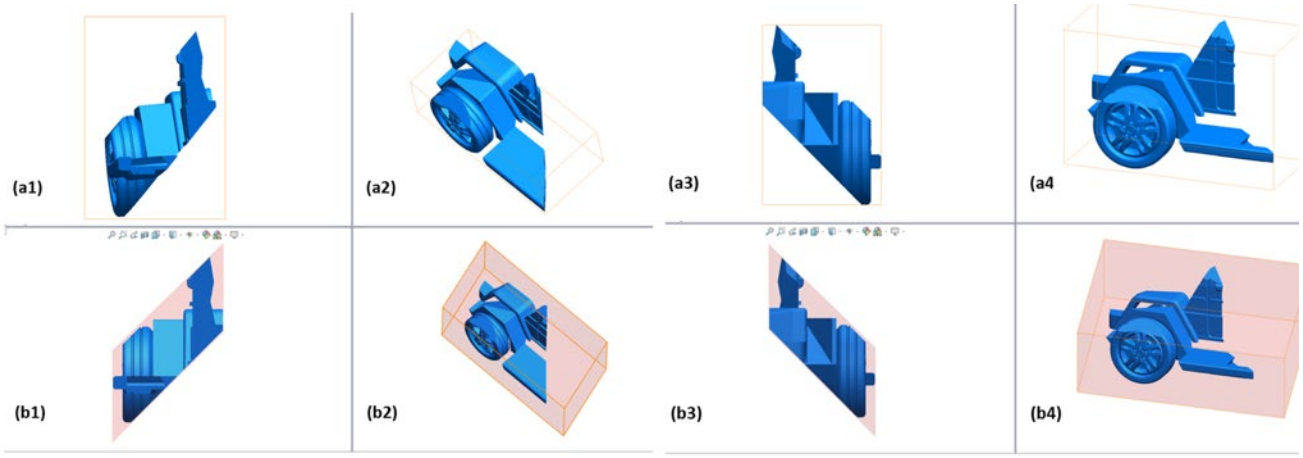

***Figure 10**: Different view of actual shape of chunk 9 from Figure 7(c) (a1) Front View (a2) Isometric view (a3) Dimetric view (a4) Back view. The representation of chunks by smallest enclosing parallelepiped with corresponding views (b1-b4)*

Table 1. *Top five print schedule generated using the algorithm in addition to the one generated using the heuristic approach (labeled "H") and their total print time*

Label	*Print Sequence {sequence number: chunk numbers}*	*Time (hrs)*
1	*{0: {0,1}, 1: {2,3}, 2: {4,12}, 3: {5,13}, 4: {6}, 5: {7}, 6: {8,14,15}, 7: {9,}, 8: {10,16}, 9: {11,17}, 10: {18,19}}*	*114.62*
2	*{0: {0,1}, 1: {2,3}, 2: {4,13}, 3: {5,12}, 4: {6,7,14}, 5: {8,15}, 6: {9,16}, 7: {10,17}, 8: {11,18}, 9: {19}}*	*104.2*
3	*{0: {0,1}, 1: {2,3}, 2: {4,5,13}, 3: {6,12}, 4: {7,14}, 5: {9,15}, 6: {8,16}, 7: {10,17}, 8: {11,18}, 9: {19}}*	*104.2*
4	*{0: {0,1}, 1: {2,3}, 2: {4,5}, 3: {7,12}, 4: {13}, 5: {6,14}, 6: {8,9,15,16}, 7: {11,17}, 8: {18}, 9: {19}}*	*104.2*
5	*0: {0,1}, 1: {2,3}, 2: {4,5,12}, 3: {7,13}, 4: {6,16}, 5: {6,14}, 6: {9,15}, 7: {8,11,17}, 8: {10,18}, 9: {19}}*	*104.2*
H	*0: {0,1}, 1: {2,3}, 2: {4,5,12,13}, 3: {6,7,14,15}, 4: {8,9,16,17}, 5: {10,11,18,19}, 6: {8,9,15,16}}*	*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, the second element represents the sequence of Chunk-1 and the third element represents the sequence number of Chunk 3 and so on, giving us a total of 20 elements. For example, for print sequence labeled 1, the first two elements are both 0, which means Chunk-0 and Chunk-1 are printed together during the first print sequence (labeled 0). The third and fourth elements are both 1, which means Chunk-2 and Chunk-3 are printed during the second print sequence (labeled 1). The fifth element is 2, which means Chunk-4 is printed during the third print sequence (labeled 2). This process goes on until the end of the list.

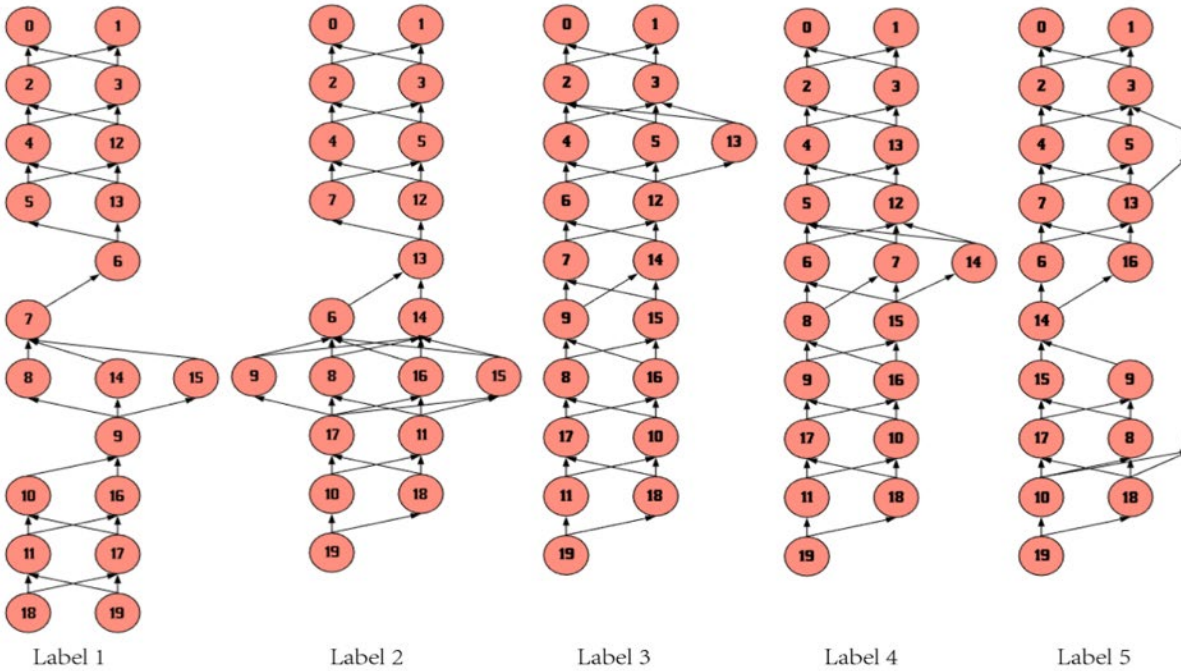

Figure 11*: The dependency tree associated with the print sequence represented in Table 1*

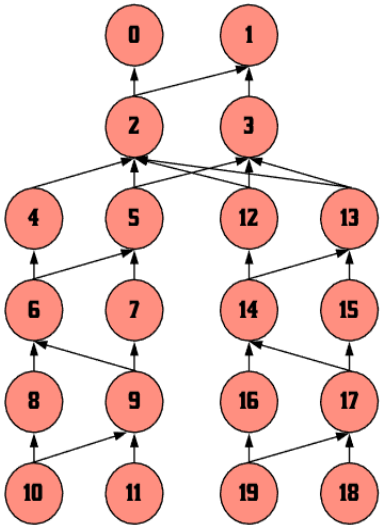

***Figure 12**: The dependency tree associated with the print sequence developed using heuristic approach*

The top five generated print schedules are presented in DDT format in **Figure 11** and the print sequence developed using a heuristic approach is presented in **Figure 12.**

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 in a stacked Venn where each circle represents a range of print time. The print schedules with longer print times are plotted in the larger circle and the ones with shorter print schedules are plotted in the smaller circles. The plot is presented in **Figure 13**. The total print time of the heuristic approach SPAR3 is also plotted in the graph for comparison, as marked by

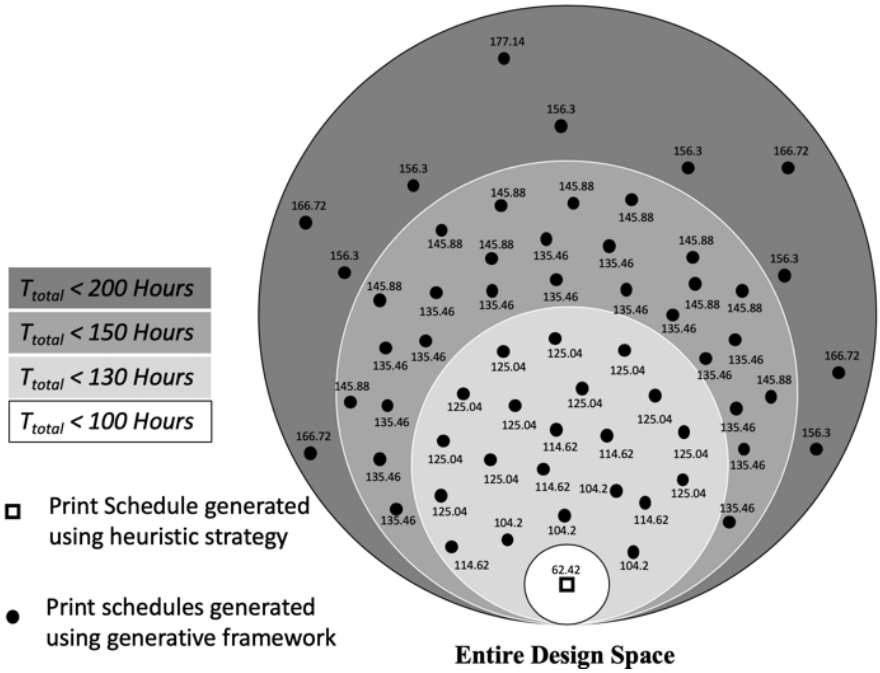

***Figure 13**: The total print time for each valid generated tree and the total print time for schedule generated using heuristic approach*

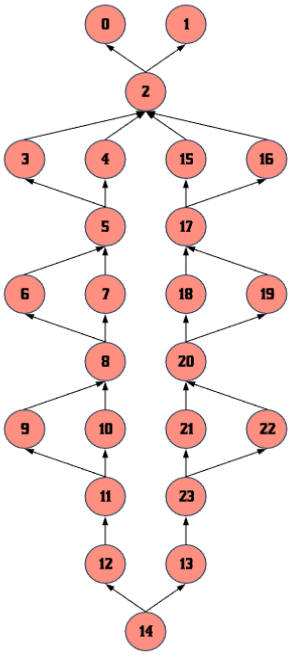

***Figure 14**: The dependency tree generated by the heuristic approach for case study II*

the square marker. As can be seen in the figure, the heuristic print schedule has shorter print time, but the generative approach was able to generate diverse print schedules.

5.2 Case Study II

In the second case with the miniature SUV, unlike the first case study, the shape of the individual chunk is different, resulting in non-uniform chunks in size and volume. As a result, the printing schedule is not as parallelized as that in case study I. The DDT representing the heuristic strategy SPAR3 is presented in **Figure 14**. It can be observed that during the majority of print sequences (i.e., different layers of the DDT), only half the number of available robots are utilized compared to maximum utilization of available robots in the first case study. This is because the physical geometrical constraints prevent maximal parallel printing. Based on this, we expect the generator to generate print schedules with less parallel print sequences as well.

Similar to the first case study, 1000 simulations were conducted based on the generative approach. This produced 50 valid print schedules. Five schedules with the shortest print time are presented in **Table 2**. The same five print schedules highlighted in **Table 2** are presented in DDT format in **Figure 15.** As expected, the generator resulted in print schedules with much less parallel print sequences.

***Table 2.** Top five print schedule generated using the algorithm in addition to the one generated using the heuristic approach (labeled "H") and their total print time*

Label	*Print Sequence {sequence number: chunk numbers}*	*Time (Hours)*
1	*{0: {0,1}, 1: {2}, 2: {3,4}, 3: {5}, 4: {6}, 5: {7}, 6: {8}, 7: {9}, 8: {10}, 9: {11,15}, 10: {12,13}, 11: {14}, 12: {16}, 13: {17}, 14: {18}, 15: {19}, 16: {20}, 17: {21}, 18: {22}, 19: {23}}*	*197.29*

2	*{0: {0,1}, 1: {2}, 2: {3,4}, 3: {5}, 4: {6,7}, 5: {8}, 6: {9}, 7: {10}, 8: {11}, 9: {12}, 10: {13}, 11: {14}, 12: {15}, 13: {16}, 14: {17}, 15: {18,19}, 16: {20}, 17: {21,22}, 18: {23}}*	*195.64*
3	*{0: {0,1}, 1: {2}, 2: {3,4}, 3: {5}, 4: {6}, 5: {7}, 6: {8}, 7: {9}, 8: {10}, 9: {11}, 10: {12}, 11: {13,15}, 12: {14}, 13: {16}, 14: {17}, 15: {18,19}, 16: {20}, 17: {21,22}, 18: {23}}*	*192.72*
4	*{0: {0,1}, 1: {2}, 2: {3,4}, 3: {5}, 4: {6,7}, 5: {8}, 6: {9}, 7: {10}, 8: {11}, 9: {12}, 10: {13}, 11: {14}, 12: {15}, 13: {16}, 14: {17}, 15: {18,19}, 16: {20}, 17: {21,22}, 18: {23}}*	*189.39*
5	*{0: {0,1}, 1: {2}, 2: {3}, 3: {4}, 4: {5}, 5: {6}, 6: {7}, 7: {8}, 8: {9,10}, 9: {11}, 10: {12,13}, 11: {14}, 12: {15,16}, 13: {17}, 14: {18,19}, 15: {20}, 16: {21,22}, 17: {23}}*	*187.31*
H	*{0: {0,1}, 1: {2}, 2: {3,4,15,16}, 3: {5,17}, 4: {6,7,18,19}, 5: {8,20}, 6: {9,10,21,22}, 7: {11,23}, 8: {12,13}, 9:{14}}*	*123.30*

The total print time for the print schedules generated using the algorithm is much longer (~1.67 times longer for the case study I and ~1.52 times longer for the case study II) compared to the ones generated heuristically with SPAR3. While the heuristically generated print schedule utilizes most of the available printing robots (two while printing the initial chunks in the center row and four afterward), the automatically generated schedules only utilize two or three at a time leaving many spare printers unused. In the first case study, the discrepancy seems larger than the second case, but the second case study represents a more practical scenario where the volume of the chunks is non-uniform leading to different printing times. In such cases, it is important to not only use the maximum number of available robots for printing but also to schedule the printing in such a way that chunks with uniform volume are printed together without violating constraints to reduce the total print time.

It is worth pointing out that although the total print time of the heuristic approach using SPAR3 was faster than that of the strategies generated by the algorithm based on the generative approach we presented, **it is expected because the goal of the algorithm is not to find the optimal printing strategy, but simply be able to generate different valid printing strategies automatically, which is an important stepping stone for the development of optimization methods for the scheduling strategies**.

Similarly, we plot the total print time of every generated valid print schedules as well as the one generated from the heuristic approach for the second case study in stacked Venn diagram, as presented in **Figure 16**.

6. CONCLUSION AND FUTURE WORK

In this paper, a generative approach is presented for automatically generating different valid printing schedules for cooperative 3D printing (C3DP) for a given number of chunks and a 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 validated printing strategy is then evaluated to calculate the total print time,

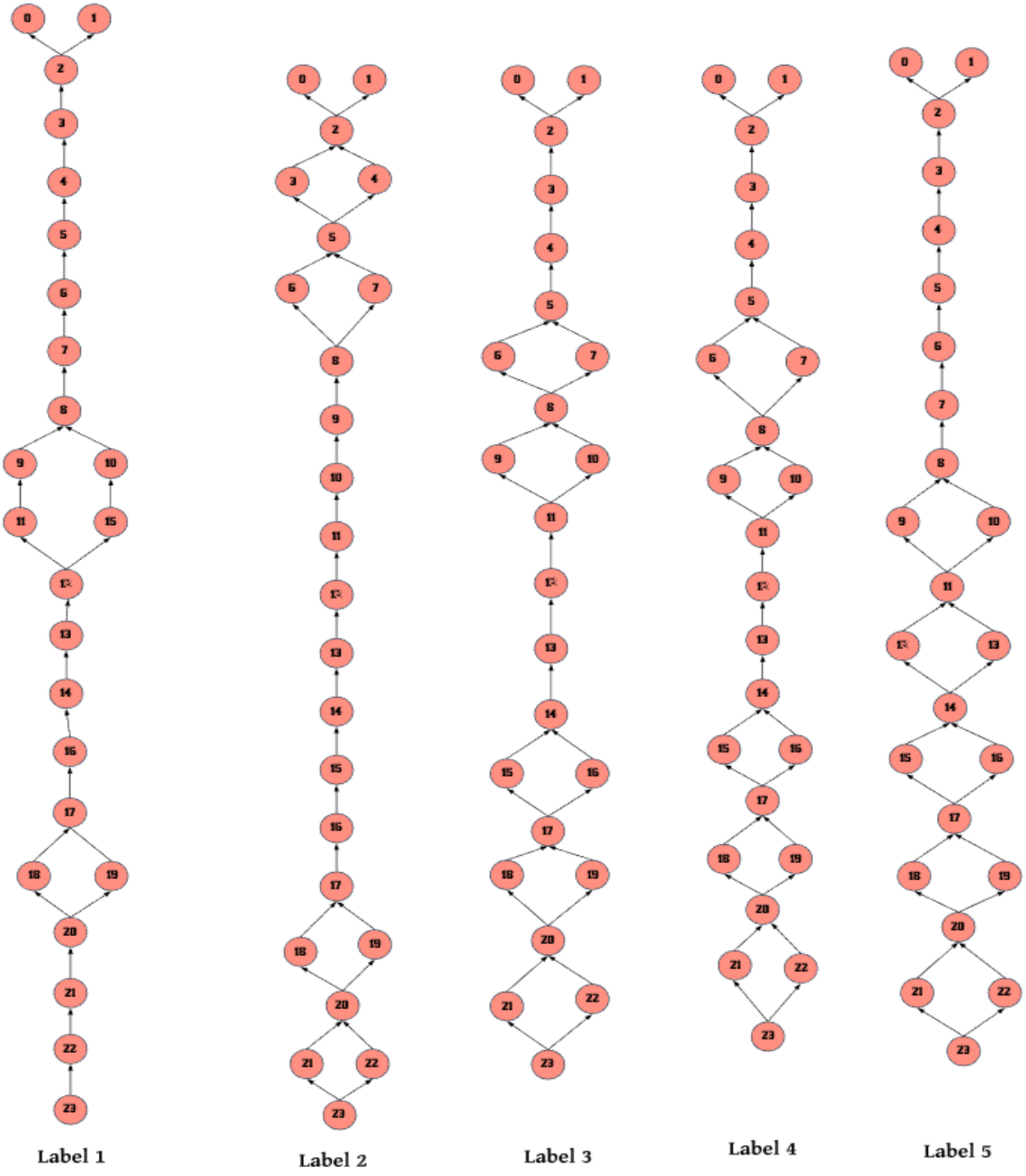

***Figure 15**: The dependency tree associated with the print sequence represented in Table 2*

using the time metrics developed. This generative approach was demonstrated using two case studies. The first case study was a large rectangular bar with simple geometry that was chunked into twenty uniform chunks with equal volume. The second case study was a more complex geometric model of a miniature folding SUV that was chunked into twenty-four chunks with different volumes. The two case studies showcase the generality of the generative approach in handling objects with different levels of complexity and scale. The key contributions of this study are:

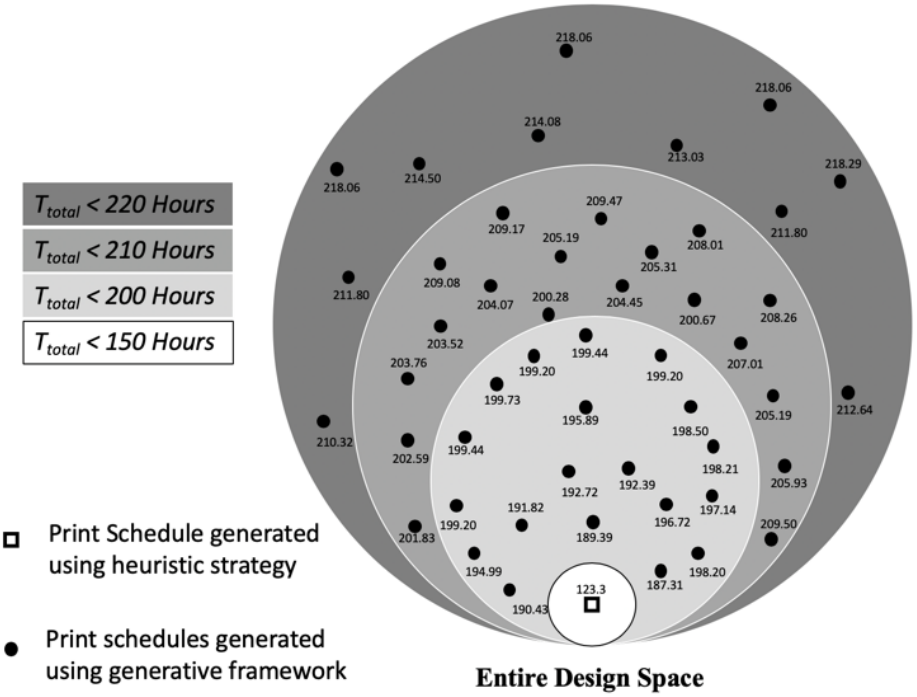

***Figure 16**: The total print time for each valid generated tree and the total print time for schedule generated using heuristic approach*

- Development of a print sequence generator that can automatically generate different print schedules by traversing the larger portion of design space that cannot be explored by human heuristics for the specified number of chunks and the available number of robots using the output of chunker.
- Use of geometric constraints identified in our previous studies to check the validity of the generated print schedules.
- Use of evaluation time metric using a directed dependency tree (DDT) developed in our previous study to determine the total print time for each valid print schedule.
- Development of a generative approach 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.
- The approach presented in this paper can be used to solve problems that couple production scheduling (IPPS) and pathfinding (MRS) with geometric partitioning (chunking) and the changing geometric constraint both temporally and spatially. Thus, the approach fills the gap in knowledge that exists in the literature of both IPPS and MRS.

Since the approach is more of an exploratory algorithm to search the entire design space, the major limitation of the approach is that the generated schedules are only guaranteed to be valid, but not optimal. In future work, we will develop an optimization approach on top of the generative approach to find an optimal collision-free print schedule with the shortest printing time. Incentives will be incorporated to promote higher utilization of available robots for maximum parallel printing. The current approach of representing the chunks with more regular geometrical shapes to relieve

computational stress will be examined in future work. Different avenues to represent the chunks could be taken and the outcomes of the approaches compared.

ACKNOWLEDGMENT

The authors would like to thank Jace McPherson for his contribution in helping us create multi-robot print simulations for C3DP. This project is supported by the National Science Foundation (NSF) Division of IIP through Grant # 1914249 and the commercialization fund through The Office of Economic Development from the University of Arkansas.

REFERENCES

[1] Love, L. J., "Utility of big area additive manufacturing (BAAM) for the rapid manufacture of customized electric vehicles." No. ORNL/TM-2014/607. Oak Ridge National Lab. (ORNL), Oak Ridge, TN (United States). Manufacturing Demonstration Facility (MDF), 2015.

[2] "Autodesk's *Project Escher"*, 2016, Accessed April 10, 2019, http://projectescher.com/,

[3] Jin, Y., Pierson, H.A. and Liao, H., "Toolpath allocation and scheduling for concurrent fused filament fabrication with multiple extruders." *IISE Transactions* 51, no. 2 (2019): 192-208.

[4] Marques, L.G., Williams, R.A. and Zhou, W., "A Mobile 3D Printer for Cooperative 3D Printing." In *Mobile Printer Design, Solid Freeform Fabrication Symposium*, pp. 1645-1660. 2017.

[5] McPherson, Jace and Zhou, W., "A Chunk-based Slicer for Cooperative 3D Printing," *Rapid Prototyp. J.*, *Rapid Prototyping Journal*, 24(9), pp.1436-1446. 10.1108/RPJ-07-2017-0150Google ScholarCrossref

[6] Poudel, L., Sha, Z. and Zhou, W., "Mechanical strength of chunk-based printed parts for cooperative 3D printing." *Procedia Manufacturing* 26 (2018): 962-972.

[7] Zhang, Z., Poudel, L., Sha, Z., Zhou, W. and Wu, D., "Data-Driven Predictive Modeling of Tensile Behavior of Parts Fabricated by Cooperative 3D Printing." *Journal of Computing and Information Science in Engineering* 20, no. 2 (2020)

[8] Poudel, L., Blair, C., McPherson, J., Sha, Z. and Zhou, W., "A Heuristic Scaling Strategy for Multi-Robot Cooperative Three-Dimensional Printing." *Journal of Computing and Information Science in Engineering* 20, no. 4 (2020).

[9] Poudel, L., Marques, L.G., Williams, R.A., Hyden, Z., Guerra, P., Fowler, O.L., Moquin, S.J., Sha, Z., and Zhou, W., Architecting the Cooperative 3d Printing System, IDETC 2020, St. Louis, MO, USA. (accepted).

[10] Kis, Tamas. "Job-shop scheduling with processing alternatives." *European Journal of Operational Research* 151, no. 2 (2003): 307-332.

[11] Petrović, M., Vuković, N., Mitić, M., and Miljković, Z., "Integration of process planning and scheduling using chaotic particle swarm optimization algorithm." *Expert Systems with Applications* 64 (2016): 569-588.

[12] Chryssolouris, G., Chan, S. and Cobb W., "Decision making on the factory floor: an

integrated approach to process planning and scheduling." *Robotics and Computer-Integrated Manufacturing* 1, no. 3-4 (1984): 315-319.

[13] Sundaram, R.M. and Fu, S.S., "Process planning and scheduling—a method of integration for productivity improvement." *Computers & Industrial Engineering* 15, no. 1-4 (1988): 296-301.

[14] Morad, N. and Zalzala, A.M.S., "Genetic algorithms in integrated process planning and scheduling." *Journal of Intelligent Manufacturing* 10, no. 2 (1999): 169-179.

[15] Shao, X., Li, X., Gao, L. and Zhang, C., "Integration of process planning and scheduling—a modified genetic algorithm-based approach." *Computers & Operations Research* 36, no. 6 (2009): 2082-2096.

[16] Li, X., Gao, L., Shao, X., Zhang, C. and Wang, C., "Mathematical modeling and evolutionary algorithm-based approach for integrated process planning and scheduling." *Computers & Operations Research* 37, no. 4 (2010): 656-667.

[17] Leung, C.W., Wong, T.N., Mak, K.L. and Fung, R.Y., "Integrated process planning and scheduling by an agent-based ant colony optimization." *Computers & Industrial Engineering* 59, no. 1 (2010): 166-180.

[18] Koes, M., Nourbakhsh, I., Sycara, K., Koes, M., Sycara, K., Nourbakhsh, I., Koes, M., Nourbakhsh, I., Sycara, K., Ramchurn, S.D. and Jennings, N.R., "Heterogeneous multirobot coordination with spatial and temporal constraints." In *AAAI*, vol. 5, pp. 1292-1297. 2005.

[19] Mishra, N., Choudhury, B.B. and Biswal, B.B., "An effective technique for generation of assembly sequence in multi-robotic assembly cells." In *2012 National conference on computing and communication systems*, pp. 1-5. IEEE, 2012.

[20] Park, J.H. and Chung, M.J., "Automatic generation of assembly sequences for multi-robot workcell." *Robotics and Computer-integrated manufacturing* 10, no. 5 (1993): 355-363.

[21] Gombolay, M.C., Wilcox, R.J. and Shah, J.A., "Fast scheduling of robot teams performing tasks with temporospatial constraints." *IEEE Transactions on Robotics* 34, no. 1 (2018): 220-239.

[22] Wan, Weiwei, Kensuke Harada, and Kazuyuki Nagata., Wan, W., Harada, K. and Nagata, K., "Assembly sequence planning for motion planning," *Assem. Autom.*, vol. 38, no. 2 (2018), pp. 195–206.

[23] Shriyam, S. and Gupta, S.K., "Task Assignment and Scheduling for Mobile Robot Teams." In *ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2018.

[24] Bhatt, P.M., Malhan, R.K., Rajendran, P. and Gupta, S.K., "Building free-form thin shell parts using supportless extrusion-based additive manufacturing." *Additive Manufacturing* 32 (2020): 101003.

[25] Bhatt, P.M., Malhan, R.K., Shembekar, A.V., Yoon, Y.J. and Gupta, S.K., "Expanding capabilities of additive manufacturing through use of robotics technologies: A survey." *Additive Manufacturing* (2019): 100933.

[26] Erdős, P. and Rényi, A., "On the evolution of random graphs." *Publ. Math. Inst. Hung. Acad. Sci* 5, no. 1 (1960): 17-60.

[27] Cordeiro, D., Mounié, G., Perarnau, S., Trystram, D., Vincent, J.M. and Wagner, F., "Random graph generation for scheduling simulations." 2010.

[28] Aho, A.V., Garey, M.R. and Ullman, J.D., "The transitive reduction of a directed graph." *SIAM Journal on Computing* 1, no. 2 (1972): 131-137.

List of Tables

List of Figures

Figure 15: The dependency tree associated with the print sequence represented in Table 2

Figure 16: The total print time for each valid generated tree and the total print time for schedule generated using the heuristic approach