Computational Frameworks for Multi-Robot Cooperative 3D Printing and Planning

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering with a concentration in Mechanical Engineering

by

Laxmi Prasad Poudel
University of Arkansas
Bachelor of Science in Mechanical Engineering, 2014

July 2021
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council.

Zhenghui Sha, Ph.D.
Dissertation Director

Wenchao Zhou, Ph.D.
Committee member

Haitao Liao, Ph.D.
Committee member

David Jensen, Ph.D.
Committee member

Sarah Nurre Pinkley, Ph.D.
Committee member

ABSTRACT

This dissertation proposes a novel cooperative 3D printing (C3DP) approach for multi-robot additive manufacturing (AM) and presents scheduling and planning strategies that enable multi-robot cooperation in the manufacturing environment. C3DP is the first step towards achieving the overarching goal of swarm manufacturing (SM). SM is a paradigm for distributed manufacturing that envisions networks of micro-factories, each of which employs thousands of mobile robots that can manufacture different products on demand. SM breaks down the complicated supply chain used to deliver a product from a large production facility from one part of the world to another. Instead, it establishes a network of geographically distributed micro-factories that can manufacture the product at a smaller scale without increasing the cost.

In C3DP, many printhead-carrying mobile robots work together to print a single part cooperatively. While it holds the promise to mitigate issues associated with gantry-based 3D printers, such as lack of scalability in print size and print speed, its realization is challenging because existing studies in the relevant literature do not address the fundamental issues in C3DP that stem from the amalgamation of the mobile nature of the robots, and continuous nature of the manufacturing tasks.

To address this challenge, this dissertation asks two fundamental research questions: RQ1) How can the traditional 3D printing process be transformed to enable multi-robot cooperative AM? RQ2) How can cooperative manufacturing planning be realized in the presence of inherent uncertainties in AM and constraints that are dynamic in both space and time?

To answer RQ1, we discretize the process of 3D printing into multiple stages. These stages in-

clude chunking (dividing a part into smaller chunks), scheduling (assigning chunks to robots and generating print sequences), and path and motion planning. To test the viability of the approach, we conducted a study on the tensile strength of chunk-based parts to examine their mechanical integrity. The study demonstrates that the chunk-based part can be as strong as the conventionally 3D-printed part. Next, we present different computational frameworks to address scheduling issues in C3DP. These include the development of 1) the world-first working strategy for C3DP, 2) a framework for automatic print schedule generation, evaluation, and validation, and 3) a resource-constrained scheduling approach for C3DP that uses a meta-heuristic approach such as a modified Genetic Algorithm (MGA) and a new algorithm that uses a constraint-satisficing approach to obtain collision-free print schedules for C3DP. To answer RQ2, a multi-robot decentralized approach based on a simple set of rules is used to plan for C3DP. The approach is resilient to uncertainties such as variation in printing times and can even outperform the centralized approach that uses MGA with a conflict-based search for large-scale problems.

By answering these two fundamental questions, the central objective of the research project to establish computational frameworks to enable multi-robot cooperative manufacturing was achieved. The search for answers to the RQs led to the development of novel concepts that can be used not only in C3DP, but many other manufacturing tasks, in general, requiring cooperation among multiple robots.

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Dr. Zhenghui Sha, who has provided me constant feedback, resources when needed, patience, and abundant support and has helped me develop both personally and professionally. He has been a great mentor and has inspired me more than anything else during my Ph.D. journey with his work ethic, knowledge, and his drive. I would also like to thank Dr. Wenchao Zhou, who has been ever-present during my Ph.D. journey and has helped me not just with new ideas development but also with his inspirational stories. Working with two of these great role models has helped me become a better thinker and learn what it means to be a good researcher as well as a good person. I would also like to show my appreciation to Dr. David Jensen and the rest of the SIDI lab members (Molla, Xingang, Yinshuang, Jared, John) as well as CAESR lab members (Charlie, Jonathan, Marvin) with whom I have enjoyed having a discussion during our weekly meetings. The AMBOTS team (Pablo, Zac, Luke) has been beneficial in generating new ideas for the swarm manufacturing platform.

During the long research hours at ENRC, the mini breaks with Edi, Chao, Jeremy were very helpful distractions and very revitalizing. In addition to this, I had the privilege to work with very talented students like Sai, Jace, Chandler, Bofan, and the development of the project would not have been possible without them.

No amount of acknowledgment would appreciate the amount of work Coral Perez has put into reading my papers and helping me improve my writing skills. Most of all, the emotional support she has provided me throughout my Ph.D. career cannot be expressed in words. She probably would have liked an honorary degree for all her work, but a paragraph

in this section has to do for now. My parents, Paramananda and Keshari Joshi, have always supported me and encouraged me to explore new things since I was a child. My brother and my uncle who were always there for me in my time of need. Pi and Chanda, my two pooches, also have been very helpful in this journey with their unconditional love.

I am also thankful to the University of Arkansas, the Department of Mechanical engineering, and the support staff, who were very patient in answering all my questions and helped me navigate the system. I would like to acknowledge the financial support provided by National Science Foundation, University of Arkansas, and the Department of Mechanical Engineering that allowed me to work on this project for my Ph.D. dissertation. Finally, I would like to thank my committee members, Dr. Haitio Liao and Dr. Sarah Nurre Pinkley, for evaluating my work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Motivation

While increasingly connected and automated, modern factories are still largely designed based on the philosophy of an assembly line, first implemented by Henry Ford in 1913. Over the past decades, the manufacturing system has evolved significantly to meet several critical requirements, including shorter lead time, greater varieties, lower prices, and flexible production to accommodate fluctuating market demands [1, 2, 3, 4]. These demands have led to new manufacturing paradigms such as flexible manufacturing systems (FMS) [5], reconfigurable manufacturing systems (RMS) [6, 7], cloud-based manufacturing (CBMS) [8], etc. While these manufacturing paradigms have greatly reshaped modern factories, the factories are still primarily designed to produce a specific product or a few specific products for an extended period. While the factory may be reconfigured to produce a different product, which often takes a significant amount of manual work and time to reconfigure and recalibrate the manufacturing machines in the factory, its configuration is typically not changed during its production time. For a specific configuration, the factory is typically designed with three constant parameters: product dimensions, production capacity, and the set of the manufacturing processes included. Although this works well for infrequent changes of production demands, it relies on the economy of scale to distribute the overhead cost of factory reconfiguration to many units and lower the average manufacturing cost per unit. This invariably leads to a concentration of manufacturing and potential congestion and disruption in the supply chain, especially when some critical factories or transportation routes in the

supply chain are forced to shut down during a regional or global crisis like COVID-19. One approach to avoid these problems in the future, is to develop a more robust supply chain with added redundancy in logistics. On the other hand, a much simpler approach could be adopted that takes inspiration from nature.

Nature has demonstrated the potentiality of the swarm through honeybees, ant colonies, and schools of fish, where hundreds or thousands of these individual agents work collectively towards a common goal. Via this cooperation, they can accomplish tasks that are nonviable by their individual counterparts, such as building huge structures (that are sometimes 1000X their size) and transporting large objects. While such demonstration has been prevalent in nature for ages, the use of large robots for cooperatively accomplishing tasks in human society continues to be challenging. Although the robotic system, these days, is getting increasingly complicated, the individual robots are designed to work by themselves and not as a part of a team in most cases in manufacturing. But as we have observed in nature, a multi-agent system can be more sustainable and more fault-tolerant, robust, and more flexible than a single-agent system. Although the benefits are apparent, the application of swarm cooperation in manufacturing has not received significant interest. However, over the last decade, there have been some seminal works in the field of construction and foraging tasks that draw inspiration from such swarm behavior in nature, such as termites to build complicated structures using a large number of simple robots [9], Slimebot, which take inspiration from slime molds and can collectively change shape to navigate through obstacles [10], Swarm-bots, that was inspired by ants and can form foraging paths [11, 12]. These research works demonstrate the natural progression of development in the research field of swarm robotics. However, the application of such research in the manufacturing set-

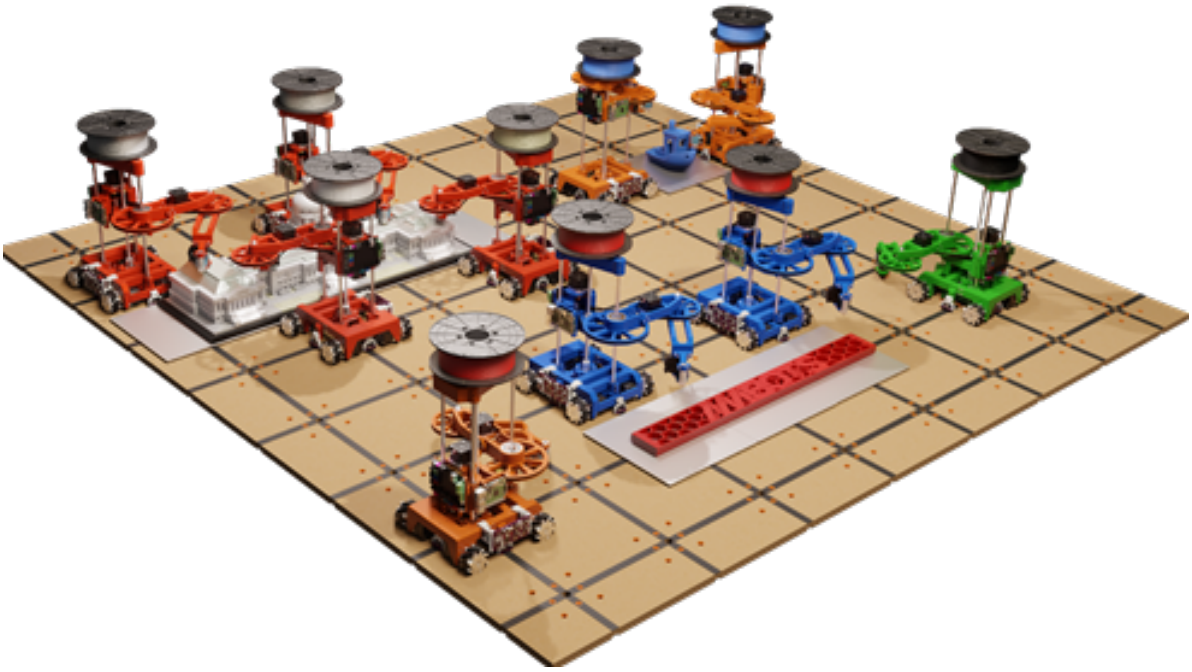

Figure 1.1: Demonstration of swarm manufacturing.

ting is lacking. The addition of the manufacturing process presents different sets of problems compared to foraging tasks and construction (moving blocks from one location to another or stacking blocks on top of one another). In manufacturing, a new material is created from an existing material by manipulating its shape and size, which creates another layer of interaction between the agents and the material it is manipulating. For example, in FDM printing, a raw material is taken and melted and reshaped into completely different structures. The manufacturing agent (FDM printers in this case) deposits one layer of material at a time after melting the raw material and continues to do so until the final part is created. Therefore, the shape and the size of the product are changing at every instant. Thus, there is a clear lack of a paradigm that utilizes the swarm mentality to manufacturing. While the application of multi-robot systems is prevalent in multi-robot research studies, they are fundamentally different in the sense that they are limited to non-manufacturing tasks, where predefined

sequences of steps are to be taken to complete the job (e.g., warehouse application, assembly operations, etc.). Therefore, while these robots might have mobility and ability to cooperate, they lack autonomy (ability to make an independent decision) in the manufacturing environment.

In my dissertation, to address this gap, I present swarm manufacturing, illustrated in Figure 1.1. Swarm manufacturing (SM) is a new paradigm for distributed manufacturing, where each factory employs thousands of mobile manufacturing robots that can regroup and recalibrate themselves in real-time. SM breaks down the complicated supply chain that is used to deliver a product from a large production facility from one part of the world to another and instead establishes a network of micro-factories at different geographic locations that can manufacture at small-scale.

In this paradigm, the product dimensions, the production capacity, and the set of manufacturing processes, which typically remain unchanged in modern factories for a specific production configuration, can all be changed in real-time by adding more manufacturing robots or regrouping the robots with software. Due to the real-time reconfiguration of the manufacturing robots, the production overhead is negligible; and thus, the average manufacturing cost per unit is relatively insensitive to the production quantity. The production scale can be achieved by distributing them across a global network of thousands of swarm-manufacturing powered factories based on regional demands. While being a promising alternative to existing manufacturing paradigms, swarm manufacturing has not been realized so far. Cooperative 3D printing, a primitive form of swarm manufacturing, where manufacturing robots are primarily 3D printing robots, has presented itself as a very good first step towards swarm manufacturing.

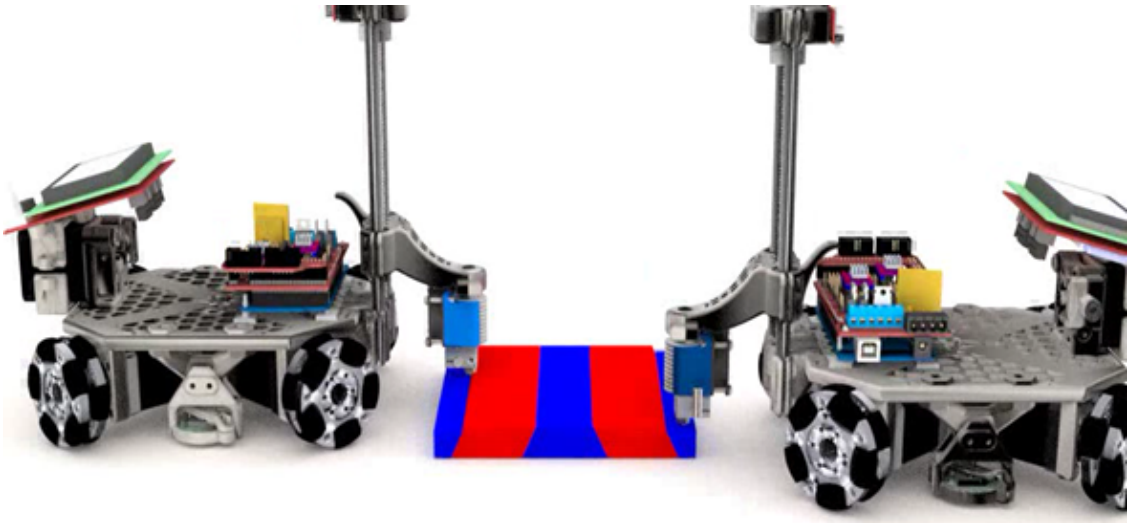

Figure 1.2: Illustration of chunk-based C3DP with two mobile 3D printers.

Cooperative 3D printing is a novel approach, inspired by nature, emulating a swarm of bees working together to build a beehive that envisions a swarm of printhead-carrying mobile robots working together to print a large object and holds the promise to provide a scalable solution to 3D printing. In C3DP, a part is first divided into multiple chunks using a chunking strategy, and these chunks are assigned to individual robots for printing. The chunking strategy is devised in such a way that no post-processing such as gluing the chunks is required, and multiple chunks can be simultaneously printed. 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 1.2. Such an approach keeps the printing local with a small chunk size, i.e., keeping the cross-section of the chunk layer small. Therefore, many printing robots can be employed to print a large part while achieving the print quality of a printer with a fine nozzle. Since the printing can be parallelized, i.e., multiple chunks can be printed simultaneously, the total print time can be significantly reduced. This new approach to 3D printing circumvents the limitations of conventional 3D printers, such as lack of scalability

and slow printing speed due to their "box" design.

1.2 Research Hypothesis, Questions, and Objectives

The overarching goal of this project was to develop computational frameworks for carrying out additive manufacturing using multiple mobile 3D printing robots. In addition to this, we want to test our *central hypothesis that the multi-robot cooperative 3D printing complements traditional gantry-based 3D printing and can provide better solutions for manufacturing in terms of flexibility, scalability, and efficiency.* The primary objective will be achieved by pursuing answers to the following specific research questions.

- **RQ1**: How can the traditional 3D printing process be transformed to enable multi-robot cooperative additive manufacturing?
- **RQ2**: How can cooperative manufacturing planning be realized in the presence of inherent uncertainties in AM and spatiotemporal constraints that are dynamic in both space and time?

Table 1.1 provides an overview of our central research objective, central hypothesis, and research approaches. On the other hand, Table 1.2 provides individual research questions, hypothesis associated with each of the questions. In addition to this, the table also contains the knowledge, that is the byproduct of the research and chapters that address the research questions.

Table 1.1: Central research objective, hypothesis, and research approaches

Central Research Objective	To establish a computational framework to enable multi-robot cooperative additive manufacturing
Central Hypothesis	The Multi-robot Cooperative 3D printing complements traditional gantry-based 3D printing and provides better printing solutions in terms of flexibility, scalability and efficiency
Research Approaches	The continuous process of 3D printing can be carried out using multiple robots by: 1. Discretizing the continuous process of 3D printing into multiple interdependent stages such as chunking, scheduling, path planning, and motion planning 2. Dividing a part into multiple chunks using a sloped-surface chunking strategy without compromising the mechanical integrity of the printed part 3. Development of computational frameworks and algorithms for multi-robot scheduling in manufacturing setting with spatiotemporally dynamic constraints 4. Implementation of decentralized multi-robot planning in additive manufacturing setting, based on simple set of rules

1.3 Research Approach Overview

Although multi-robot C3DP demonstrates excellent prospects for the future of AM, no existing method can take a digital model (e.g., an STL file) and directly plan for 3D printing using multiple robots. Thus, to achieve this objective and prove our central hypothesis, we took the following research approaches:

1. *Discretizing the continuous process of 3D printing into multiple interdependent stages such as chunking, scheduling, path planning, and motion planning (illustrated in Figure 2)*

   (a) *Chunking*: Chunking is the process of dividing a single 3D model CAD file into

Table 1.2: Detailed research questions and hypothesis

Research Questions and Hypotheses	**Research Question 1**	**Hypothesis 1**	**Chapters**	**Expected Outcomes**
	How the traditional 3D printing process can be transformed to enable multi-robot cooperative AM?	The transformation of the traditional 3D printing into a multi-stage framework of chunking, scheduling, path planning and motion planning can realize multi-robot cooperative AM	Ch. 3: Identify constraints that must be satisfied in order to generate working print strategies for C3DP Ch. 4: Explore the design space to generate diverse valid print schedules for C3DP Ch. 5: Search for optimal or near optimal print schedules while accounting for chunk assignment with limited printing resources Ch. 7: Investigate the mechanical strength of chunk-based parts and compare with that of the conventional 3D printed parts	• Knowledge of how the mechanical strength of part compares with that of the conventional printed parts • A set of constraints that needs to be satisfied in order to implement multi-robot C3DP print strategies • A generative approach that explores large design space for valid print schedules for C3DP • A scheduling framework for C3DP based on chunk assignment with limited printing resources
	Research Question 2	**Hypothesis 2**	**Chapters**	**Expected Outcomes**
	How can cooperative AM planning be realized in the presence of uncertainties and spatiotemporal constraints that are dynamic in both space and time?	The planning for multi-robot C3DP can be realized by using a decentralized approach that can handle inherent uncertainties that are difficult to predict in AM	Ch. 6: Develop a decentralized approach for planning of multi-robot C3DP Ch. 6: Compare the makespan and the runtime of decentralized approach with that of centralized planning	• A rules-based decentralized planning approach applicable in manufacturing scenarios & can handle uncertainties in AM. • Quantitative analysis to provide comparison between the centralized and decentralized planning in the context of C3DP

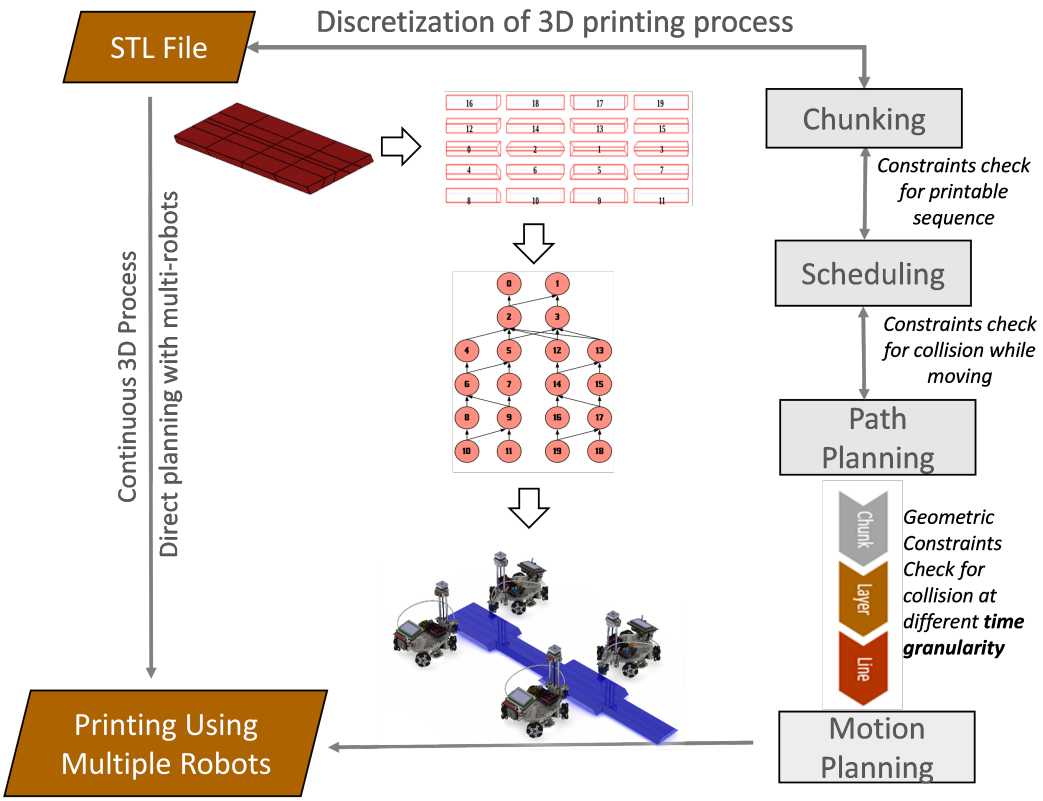

Figure 1.3: Research Overview. Multiple stages of C3DP.

multiple small pieces or chunks without compromising the quality of the final part. It is like a geometric partitioning problem with a higher level of complexity; it has an added constraint, which requires the part to be chunked so that multiple robots can work on printing individual chunks simultaneously.

(b) *Scheduling*: Once a part is divided into multiple chunks, these chunks are then assigned to individual robots for printing. Doing so falls under the umbrella of scheduling. Scheduling consists of two main aspects: chunk assignment and chunk scheduling. Chunk assignment is a process of assigning individual chunks to available robots; chunk scheduling generates a print sequence based on the dependency relationship between the chunks and the number of robots.

(c) *Path Planning and Motion Planning*: After the generation of the print schedule, path-planning and motion-planning need to be done to execute the schedule. Motion-planning comprises finding a sequence of print movements, including infill printing path and perimeter printing path, which are taken to print individual chunks. On the other hand, path planning includes finding a sequence of paths that moves a robot from the current location to the next print location. Both must be carried out while avoiding any collision (collision between the robots and collision of the robots with the printed part).

However, the discretization does not turn this continuous problem into a common discrete problem studied in existing multi-robot systems research. There are inherent inter-dependencies between multiple stages resulting from the continuous nature of the problem, and additional considerations must be taken into account. For example, while chunking requires consideration issues regarding the mechanical strength of the part, it also requires considerations such as scheduling (does the chunking method allow multi-robot printing simultaneously?), the number of robots (does chunking limits the number of robots that can be employed simultaneously?), and path planning (does the chunking result in scenarios with no feasible path?). These considerations ensure that the printing process is physically viable, and no additional post-processes such as bonding chunks using glue are required. Additionally, the geometric constraints are dynamically changing spatially and temporally, with a strong dependency on the geometry of the digital model and the chunking, scheduling, and path planning strategies. These unique challenges distinguish C3DP from existing 3D printing research.

2. *Dividing a part into multiple chunks using a sloped-surface chunking strategy without compromising the mechanical strength of the part*

   When an object is broken down into smaller chunks and printed using different printers, questions arise about the overall mechanical integrity of the final product. Since this is the first time, to the best of my knowledge, a chunk-based approach is used to print an object without requiring post-processing, a study is needed to understand what parameters impact the said integrity of chunk-based parts. Such a study can identify the parameters and allow the user to decide what values of parameters to choose to print mechanically strong chunk-based parts.

3. *Development of computational frameworks and algorithms for multi-robot scheduling in manufacturing setting with spatiotemporally dynamic constraints*

   Once a part is divided into multiple chunks, they need to be assigned to individual robots, and a print schedule needs to be generated. The generated print schedule needs to ensure collision-free printing. This requires leveraging theoretical knowledge from other fields such as mathematics, operation research and testing out existing algorithms, and exploring new ones.

4. *Implementing the decentralized multi-robot planning in AM settings, based on simple set of rules*

   Both the centralized and decentralized approaches to multi-robot planning have been widely discussed in the context of non-manufacturing tasks; however, extant literature lacks such comparison in manufacturing, more specifically additive manufacturing using multiple robots C3DP. Thus, to better understand how either of the planning

approaches performs and quantify these performances in terms of makespan and computation time, a comparative study is lacking.

1.4 Contributions

This dissertation revolves around the development of a multi-robot 3D printing system and mostly focuses on the development of computational frameworks for multi-robot scheduling in 3D printing. Thus, the primary contributions of this dissertation are:

1. **Discretization of the continuous process of 3D printing using multiple robots into multiple discrete stages such as chunking, scheduling, path planning, and motion planning**

   This is the first time, to the best of our knowledge, a 3D printing process has been discretized into multiple steps for the application of multi-robot printing. Such discretization simplifies the problem of multi-robot printing by dividing a larger problem into series of smaller problems such as chunking, scheduling of multiple robots, path planning for the robots, and motion planning to achieve the printing. While the dependencies between the different stages have to be taken into account, discretizing a larger problem into a smaller multistage problem allows a researcher to focus on one aspect of the problem at a time. Although there have been approaches in the past that have tried to implement multi-agent printing, such as Project Escher, the complexity of the problem and the stochastic nature of 3D printing resulted in the project being discontinued. Our approach, on the other hand, has shown great promise by demonstrating the printing of several large-scaled printed parts such as honeycomb artwork of dimension

$844mm \times 90mm \times 12mm$ and AMBOTS logo of dimension $1200mm \times 250mm \times 75mm$.

2. **Computational frameworks and algorithms for multi-robot scheduling in a manufacturing setting with temporospatially dynamic constraints**

   While there are many research works that have developed multi-robot scheduling, the problem addressed is discrete in nature, such as assembly operations, search and rescue, team formation, and object transportation. On the other hand, the task of 3D printing is a manufacturing task, where the material is gradually added until the final part is manufactured. Thus, the topology of the structure changes both spatially as well as temporally. We call such tasks continuous because the material is continually deposited until the final object is fully printed. No existing algorithm in related research fields has demonstrated its use for such tasks using multiple robot systems. This is the first time, to our knowledge, a multi-robot scheduling algorithm has been developed and implemented in an actual manufacturing context, using 3D printing technology. The computational frameworks and algorithms presented in chapters three, four, and five can handle the manufacturing constraints, which constantly change and allow collision-free printing using many printing robots.

3. **Implementation of decentralized planning approach in an additive manufacturing setting for multi-robot systems, based on a simple set of rules**

   Although the decentralized planning approach demonstrates superiority over the centralized approach in terms of dealing with uncertainties in manufacturing and computational burden in large-scale problems, their application has not been ubiquitous in multi-robot planning problems. Multi-robot planning is an overarching problem and

includes scheduling, path-planning, and manufacturing. Thus, it is a more complicated problem than its individual component. Such planning requires solving multiple NP-hard problems (scheduling and path planning of multiple robots) and could benefit from the use of a decentralized approach, despite its inability to guarantee theoretical optimality. To the best of our knowledge, this is the first time the application of a decentralized approach based on a simple set of rules has been demonstrated in terms of 3D printing applications using multiple mobile robots. Though the implementation of a rule-based approach was demonstrated by Werfel et al. in construction (picking and placing prefabricated blocks) [13, 14, 9], the application context is quite different, and the governing set of rules are quite different as well.

1.5 Outline and Roadmap

The dissertation is divided into seven parts, and the roadmap of the dissertation is provided at the end of the chapter.

Chapter 2. Related research This chapter discusses the different manufacturing paradigms that have been presented over the last few decades and how they differ from the swarm manufacturing paradigm presented in this dissertation. The chapter then discusses cooperative 3D printing (a form of swarm manufacturing) and some of the research works that have attempted to enable parallel printing using different methodologies. The focus is then shifted towards the most difficult problem of cooperative 3D printing, multi-robot planning for manufacturing tasks. The chapter focuses on two related research fields: multi-robot systems and integrated process planning and scheduling. The differences between the works in those research fields and the cooperative 3D printing are highlighted in this chapter. Finally, we

conclude by outlining research gaps based on the reviewed literature.

Chapter 3. Heuristic Scaling Strategy for C3DP This chapter discusses the first working print strategy for cooperative 3D printing that was developed using human heuristics. In addition to this, the chapter also presents a framework that can be used to check the validity of the print schedules by leveraging the geometry of the printed chunks and the printing robots. An evaluation criterion is presented that can be used to calculate the total time it takes the robot to print available chunks using the concept of the directed dependency tree. The validity of the generated print strategy is then tested as well as the total print time evaluated using the framework on two case studies: a rectangular bar and a topological map of the state of Arkansas.

Chapter 4. Computational Framework for C3DP In this chapter, a computational framework is presented to explore a large design space for C3DP schedule generation. Unlike the print strategy that was generated using human cognition, the framework presented in this chapter automatically generates multiple print schedules using the concept of the adjacency matrix and checks the validity of the generated print schedules using the geometric constraints presented in the previous chapter; and finally, evaluates them for total print time.

Chapter 5. Resource-Constrained Scheduling for C3DP In this chapter, a resource-constrained framework for multi-robot 3D printing is described. While the computational framework attempts to explore the entire design space to search for print strategies, the two methods presented in this chapter methodologically reduce the search space by adding constraints to their search criteria. The results obtained using the two methods are compared using different case studies that differ in scale as well as geometric complexity.

Chapter 6. Planning for Multi-Robot Manufacturing using C3DP Here, we demon-

strate the use of a decentralized approach to multi-robot manufacturing applications. We then compare the performance of the centralized approach presented in the previous chapter with the decentralized approach based on a simple set of rules. In analyzing the results, we compare the computational cost of each approach, their ability to obtain quality solutions in terms of makespan, and their performance in the presence of uncertainties.

Chapter 7. Mechanical Strength of Chunk-based Parts Printed Using C3DP This chapter provides a brief overview of the sloped-surface chunking strategy. We then analyze how the mechanical strength of the chunk-based parts compares with that of one printed using conventional 3D printing. We identify three direct parameters that impact the mechanical strength of chunk-based parts and derive the relationship between the parameters and the tensile strength of the part based on experimental data.

Chapter 8. Conclusion This chapter provides a concluding remark and major contributions of the dissertation. The research question posed at the beginning of the dissertation are revisited in this chapter to see if and how they were answered. In addition to this, it also provides suggestions regarding future directions, including the use of artificial intelligence techniques such as reinforcement learning and the search for an approach that could allow layer-level printing.

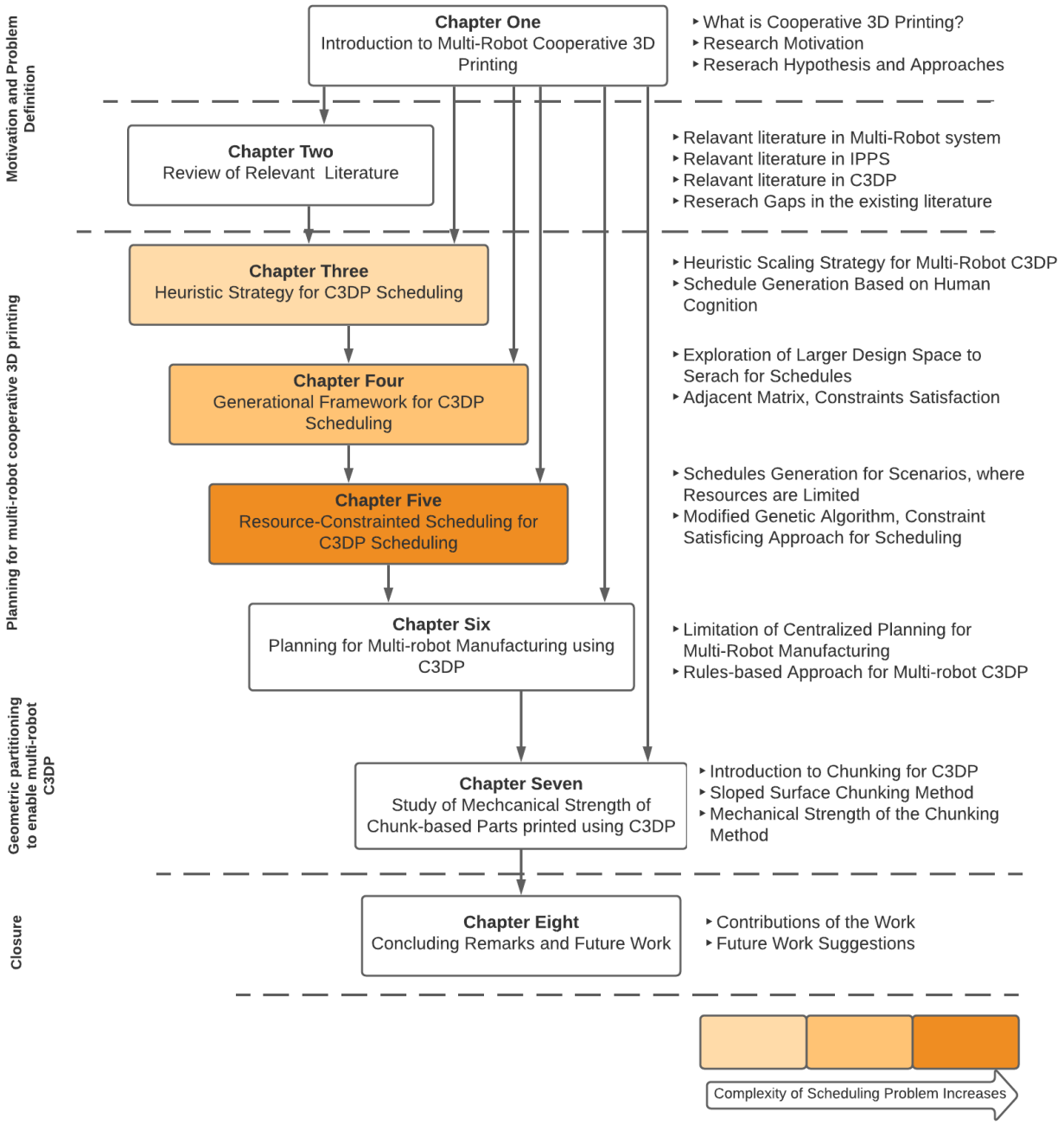

Figure 1.4: Thesis Roadmap.

2 REVIEW OF RELEVANT LITERATURE

2.1 Manufacturing Paradigm

Over the last few decades, many different manufacturing paradigms have been proposed in an attempt to evolve the traditional dedicated line-based manufacturing that solely focuses on producing one type or one family of products to meet the demand for more individually customized product variants. One such paradigm is Agile Manufacturing, whose central idea is to be resilient to continuously changing environments [15], is more of a management philosophy than a manufacturing technique [16]. It focuses on achieving flexibility and responsiveness during times when market demand is changing. Although it is not a manufacturing technique, it promotes useful approaches such as smaller, modular production factories that can produce different types of products with minimal effort. Such an approach injects a more pragmatic view to manufacturing and enables "quick response" to the changing market demand [17, 18]. Another manufacturing paradigm that has gained a lot of attention is the Flexible Manufacturing System (FMS) [19]. FMS puts a high focus on flexibility rather than on production efficiency [20]. FMS is expected to provide a good mixture of productivity and a variety of goods. Having a good combination of the two allows the production facility to be resilient to the change in customer demand. While most researchers focus on developing flexible machinery in the manufacturing plant to achieve flexibility in manufacturing, Gupta et al. argue that flexibility does not come solely from the machines, but combinations of factors such as physical conditions and characteristics, supply-chain establishments, management practices, and information integration [21]. A ro-

bust FMS should have the ability to identify and differentiate different products, a quick transition of operation instructions, and a rapid transition of physical setup [20]. A similar concept to the flexible manufacturing paradigm is called the Reconfigurable Manufacturing Paradigm (RMS). Reconfigurable manufacturing attempts to combine the advantage of flexible systems with that of dedicated line systems. RMS demonstrates higher throughput than the flexible manufacturing system but lower than the dedicated line system [22, 6]. The design of RMS is done with six-core ideas, including scalability (designed for capacity change in the future), convertibility (designed for functionality changes in the future), Diagonasibility (designed for easy problem diagnosis), customizability (designed to produce flexible parts of the product family), modularity (having modular design), integrability (designed for fast integration) [22].

Over the last few years, Cloud Manufacturing has gained a lot of attention [8]. This concept has drawn a lot of scrutiny and is even touted as one of the main directions in the development of the manufacturing industry [23]. Cloud manufacturing is an emerging paradigm whose central idea is to have networks of manufacturing factories that can provide users with manufacturing resources and capabilities in the form of on-demand services. With the new technological advances such as cloud computing, Internet of Things (IoT), distributed computing, and service-oriented architecture (SOA), users can request services for a wide range of applications such as manufacturing, product design, and testing, using the internet. Thus, this manufacturing paradigm is showing a lot of promise. The technology (swarm manufacturing) developed in this dissertation can complement the concept of cloud manufacturing, where swarm manufacturing factories can be nodes that will be part of a larger global cloud-based manufacturing network. Distributed Manufacturing (DM) is a manu-

facturing paradigm that focuses on establishing small, flexible, and scalable manufacturing units distributed geographically [24]. DM highlights the importance of growing democratization and decentralization of manufacturing and focuses on "distributed economies", which eludes to the fact that such approach would have flexible networks with local raw materials and resources and other local dimensions, resulting in environmental benefits that could lead to the sustainable manufacturing [24, 25, 26]. DM focuses on higher-level implementation, which includes establishing manufacturing networks, developing business models for operating in "distributed economies", reduction of emissions through reduced transportation, and thus, shorter supply chains. SM is one of the approaches that can enable DM. It does so by developing enabling technologies such as cooperative 3D printing that allows heterogeneous manufacturing (different product families, not limited to one product or one family of products) at factories established in the local ecosystem.

Table 2.1: Brief summary of different manufacturing paradigms

Manufacturing Paradigm	**Defining feature**
Agile Manufacturing	Resiliency during changing demand (management approach)
Flexible Manufacturing System	More variant capability: Quick transition from one production to another in time of need
Reconfigurable Manufacturing System	Low and fluctuating volumes: have both dedicated lines along with flexible systems
Cloud-based Manufacturing	Manufacturing as service on demand: establish global manufacturing networks
Distributed Manufacturing (Swarm Manufacturing)	Redistribution of manufacturing: decentralization and democratization of manufacturing

While different paradigms have been proposed to support the changing behavior of customers, an overarching theme can be observed in all these systems: that the manufacturing system needs to adapt to the changing demand of customers without significant economic

impact. This means breaking down a larger value chain into multiple parts and processes and distributing manufacturing across different geographical locations. While production efficiency was the only priority in the past, when products had to be mass-produced, the market demand has changed. Thus, flexibility must be made a priority, even if it means making a slight compromise in production efficiency. The increase in demand for individualized or personalized product variants requires on-demand, smaller-scale, localized manufacturing capable of producing different types of products. This is the vision of swarm manufacturing, where something as simple as a chaise lounge can be manufactured alongside a vacuum cleaner. Swarm manufacturing envisions multiple heterogeneous mobile robots working together to manufacture the desired product alongside other robots that might be working on producing something completely unrelated. Such an approach to manufacturing has no substantial cost of reconfiguration as it can be done in real-time. This, complemented with the cloud manufacturing networks of such swarm manufacturing factories, can support the changing demand of customers. Brief defining features of the presented manufacturing paradigms are presented in Table 2.1.

Through cooperative 3D printing, we have demonstrated the applicability of swarm manufacturing using 3D printing technology. Cooperative 3D printing is a basic form of SM, where collaborating robots primarily do 3D printing. C3DP is the first step towards the implementation of SM. Though only a first step, the application of C3DP is quite challenging as it requires cooperation between the multiple 3D printing robots in shared space while undertaking manufacturing tasks. While this is just the beginning, we believe the future of manufacturing is heading towards establishing micro-factories where hundreds of robots work together to manufacture different types of products but are coordinated and connected

to scale up or scale down productions based on customer demand. In the next section, we discuss some of the other research works that have attempted to realize cooperative 3D printing.

2.2 Existing work on Cooperative 3D Printing

While the approach presented in this dissertation allows multiple mobile 3D printing robots to enable parallelism and cooperation between the robots, several other approaches have been attempted both in the past and in ongoing research to increase the print speed and scalability of the 3D printing system. The most used 3D printing system is gantry-based. They use a gantry to position the print nozzle in XYZ cartesian coordinates [27]. The most common approach to enable parallelism and cooperation in such a gantry-based printing system includes the addition of multiple extruders to increase the print speed. For example, Jin et al. presented a computational approach for multiple printing extruders that cooperate to print an individual layer of a part based on Project Escher's hardware platform [28]. They were able to reduce each layer's printing time by as much as 60% using three extruders. While these methods can achieve a certain level of cooperation between printheads, the whole system does not scale up well with larger parts because integrating many components into a single machine creates design challenges and increases cost disproportionally. The second approach to improve the scalability of the gantry-based 3D printing system includes increasing the size of the printer itself. One of the widely renowned large-scaled gantry-based printers is BAAM (big area additive manufacturing) system developed by Oak Ridge National Lab, which is one of the world's largest electron beam-based 3D printers) [29]. Other such large 3D printing system includes architecture fabricators such as Brach Technology

and Apis Cor, that specializes in construction-scale 3D printing as well as architectural projects [30]. While these approaches increase the scale of the printable parts and increase the print speed, they demand larger layer thickness to accommodate the accuracy of current motion systems. Additionally, larger cross-sections lead to a longer waiting time between scanning paths, and therefore, a larger temperature gradient for heat-involved processes, e.g., FDM and Selective Laser Sintering [31]. This could inevitably result in high internal stresses in parts and ultimately cause part warping or failure.

More recently, cooperative 3D printing has gained significant steam because it provides scalability to 3D printing by enabling multiple mobile 3D printing robots to cooperate on a single job. The use of multiple robots provides a new approach to 3D printing, where a printing nozzle is not enclosed inside the box but is attached to individual robots as end effectors. This change of design allows the printer to be able to print a part much larger than that allowed by the previous "box" design, and multiple extruder-carrying robots can work together, improving the print speed. The increasing number of ongoing research in cooperative 3D printing demonstrates the potential future value proposition of the use of multiple robots for a single print job. A Study at Singapore's Nanyang Technological University revealed the printing of large-scale concrete structures concurrently using two mobile robots with SLAM (simultaneous localization and mapping) technology [27].

Similarly, Hongyao et al. demonstrated a large-scale 3D printing system composed of multiple robots working in collaboration [30]. They also presented a robot interference avoidance strategy by dividing the printer layer into several safe and interference areas and a scheduling algorithm based on efficiency egalitarianism. The result presented demonstrates that the use of four robots can improve the system's efficiency by more than 73% compared

to the traditional 3D printing system that uses a single printing nozzle. In addition to this, Robotic Swarm Printing (RSP) systems take their inspiration from the biological world to employ a swarm of printers for large-scale constructions [32]. Though the use of multiple mobile robots overcomes the limitation of traditional 3D printing, it also brings new sets of challenges. There is no existing approach that allows users to take a 3D model of a part and do 3D planning using multiple 3D printing robots. So, the 3D model of a part must be partitioned into multiple small "chunks" so that multiple robots can work on printing those simultaneously. However, since the advantage of C3DP lies in printing the chunks in parallel, it needs to be ensured that the result of partitioning supports such parallel printing and minimizes the amount of post-process work such as gluing the chunks together. No existing geometric partitioning studies consider such issues while dividing a part into smaller pieces. Once a part is divided into smaller chunks, these chunks need to be assigned to the available printing robots, and a schedule needs to be generated so that the robots can work on printing the assigned chunks without resulting in a collision between the active robots as well as a collision between the robots and the printed parts.

One of the most challenging aspects of cooperative 3D printing is the scheduling of multiple printing robots to achieve collision-free printing. Such challenges exist due to the innate complex-schedule constraints (e.g., task dependencies constraints) and temporalspatially coupled relationships between the task as well as the robots. Multi-robot task scheduling has been researched extensively in different research fields, including multi-robot system research (MRS), integrated process planning, and scheduling research (IPPS). While not focused explicitly on multiple robots, the research field of operations research and parallel computing research covers multiple machines working in parallel to accomplish multiple

assemblies or logistics tasks. Thus, the rest of the chapter discusses previous research related to multiple robots working on completing a cooperative task in the aforementioned research fields. In addition to this, research works related to the more specialized topics such as decentralized planning and chunking are included in chapter six and chapter seven, respectively.

2.3 Multi-Robot Systems Research Domain

Multi-robot task allocation (MRTA) and scheduling have widely been researched in the past few decades. The use of multiple robots to perform tasks allows the robots to work in parallel, and such an approach can create synergies that cannot be achieved by the use of a single robot. The problem of multi-robot task allocation and scheduling is usually categorized using the taxonomy introduced by Gerkey and Mataric [33], where they first classified robots based on their ability to perform either single or multiple tasks at a time. They then differentiated between the task that either requires a single- or multiple-robots to complete and finally differentiate between the instantaneous task allocation (tasks that are to be allocated to robots instantaneously and requires no future allocation) versus time-extended assignment (each robot is allocated multiple tasks that are to be executed based on a given schedule). Based on this taxonomy, C3DP falls under the MT-MR-TA problem, which basically translates to the multiple-tasks-multiple-robots-time-extended task assignment problem. As multiple robots are involved in accomplishing multiple tasks, the robots must take the dependencies between the tasks into account. Some of these dependencies can be formulated as precedence constraints (dependencies between the tasks) that exist between the tasks, which cannot be broken [34]. In addition to this, other constraints in MT-MR-TA

include ordering constraints that can be synchronization constraints (two tasks have to start at the same time) [35], exclusive or (XOR) constraints (only one of the tasks can start at a time).

While multitudes of research have been conducted for multiple robot task allocation, the tasks undertaken by robots in the research domain are discrete tasks. Discrete tasks are the tasks that can be solved by taking a discrete number of steps and includes tasks such as pick and place assembly, search and rescue, and pattern formation. In general, two main approaches have been used to solve task allocation and scheduling problems in MRS: centralized approaches and decentralized approaches. The centralized approaches require a central planner responsible for planning the actions of all robots and communicating with individual robots. On the other hand, the decentralized approaches involve no central planner, and the planning responsibility is distributed among all the independently operating robots that rely solely on information accessible to individual robots. The prominent differences between the two approaches are highlighted in Table 2.2. More details of the ongoing decentralized research are presented in chapter six. Centralized approaches often involve mathematical optimization, also known as exact methods, such as linear programming [36], integer programming [37], and combinatorial optimization [38]. These optimization approaches use branch and bound, branch and cut method to converge to optimal result [39]. Atay et al. used a mixed-integer linear programming (MILP) approach to allocate heterogeneous robots for detecting and controlling multiple regions of interest in an unknown environment [40]. The objective function that they used contained four different basic requirements such as control of ROIs, communication between the robots, detect ROIs, and control of the maximum area, along with some constraints such as avoiding obstacles and following the speed

limit. Similarly, Sundar and Rathinam use MILP formulation along with a heuristic to solve routing problems for fuel-constrained Unmanned Aerial Vehicle (UAV), where multiple refueling depots are available that UAV can recharge before running out of fuel [41]. While these optimization-based approaches can obtain an optimal result in a reasonably short amount of time, they do not scale well with larger scaled problems. In addition to lack of scalability, the exact methods do not work well in dynamic environments where constraints constantly change with time.

Table 2.2: Differences between the centralized approaches vs. decentralized approaches

Category	**Centralized Approach**	**Decentralized Approach**
Efficiency	Typically, more efficient and can enable globally optimized solutions	Typically, less efficient, and difficult to achieve global optimum due to distributed decision making
Communication cost	Central planner needs to be in constant contact with the entire team, which results in high communication cost, higher bandwidth	Low communication cost for local communication as local transmit information locally
Robustness	Single point of failure i.e., if central planner fails, the system fails	No single point of failure
Response to dynamic conditions	Requires replanning in dynamic environment	Individual agents respond to local environment so, very well suited for dynamic environment
Scalability	If the scale of problem increases, the computational requirement increases	Computation cost increases at a lower rate compared to centralized approaches
Quality of solution	Guarantee optimality if mathematical programming used. It is possible to achieve global optimum	No theoretical guarantee of optimality. It is difficult to achieve global optimum

Metaheuristics methods are also often used in centralized planning, such as simulated annealing [42] and genetic algorithm (GA) [43], which require less computational

cost compared to the exact methods such as linear programming. However, they cannot guarantee an optimal result and attempt to achieve near-optimal results. Population-based approaches such as genetic algorithms have been widely used to search for a feasible solution for multi-robot task allocation and scheduling problems [44]. They have successfully obtained excellent results for multi-robot task allocation problems. For example, Panchu et al. presented a multi-objective approach for MRTA with precedence constraints using GA with its parameter turned using Taguchi's design of experiments [45]. The study considered both robot-centric (total travel time by robots) and task-centric objectives (such as total task completion time). As a result, they could identify the optimal number of robots for a given set of tasks.

Similarly, Jones et al. used GA to provide a solution for task multi-robot time extended task allocation problem in simulated disaster response domain [46]. The approach presents a non-heuristic GA-based approach that provided a very high-quality solution for problems with intra-path constraints. Particle swarm optimization [47, 48], ant colony optimization [49, 50], and honeybee mating optimization [51] are other population-based optimization approaches that have been widely reported to solve MRTA problems.

2.4 Integrated Process Planning and Scheduling

The job shop problem deals with multi-operation scheduling for multiple stationary machines. This is one of the most common problems in IPPS and OR research. Such problems focus on batching and minimizing the makespan of multiple jobs arriving at different times. For example, Jin et al. presented a modified hybrid honey-bee-mating optimization approach with integrated simulated annealing to minimize the make-span of the production

process with multiple jobs and multiple static machines [52]. Similarly, Shao et al. used modified GA to minimize the make-span of a similar production process and to balance the utilization of machines [43]. The use of mathematical modeling is widely used in IPPS as well as OR research. For example, Gong et al. used mathematical modeling to remanufacturing-oriented IPPS problems [53]. Meng et al. used MILP (Mixed integer linear programming) models for energy-aware flexible job shop scheduling problems [54]. A more detailed review of the use of different mathematical models along with metaheuristic methods used in job shop scheduling problems and its perspective under industry 4.0 is presented by Zhang et al. [55].

A class of problems similar to the job shop problems is multi-robot assembly scheduling, where multiple robotic arms are used for carrying out a pre-determined sequence of assembly operations. For example, Tereshchuk et al. proposed an efficient scheduling algorithm for task allocation in assembling aircraft structures using robotic arms. The algorithm relies on the workload balancing of the tasks among the robots as well as ensures collision-free scheduling [56].

2.5 Research Gaps

Although there have been some approaches to parallelize printing of a part using multiple extruders in cooperative 3D printing to reduce the printing process's makespan, but the implementations are usually limited to gantry-based printing systems as discussed in section 2.2. The use of a large-scale gantry-based system force the user to choose between quality and scalability. They cannot provide both simultaneously without significantly impacting the total print time. The cost of the systems also could be a significant hindrance for wider

adoption, especially for large-scale applications. Thus, there is a need for a cooperative printing solution that can be scaled up and down without significant cost, overcomes the scalability issue of the traditional gantry-based printing, and can increase or decrease the number of printing agents to speed up or slow down the printing process. Cooperative 3D printing systems address this issue by using multiple mobile 3D printing robots to print a desired part in much shorter time than conventional 3D printers. Compared to other existing 3D printing methods, cooperative 3D printing prevails in many areas. First, it offers one of the most flexible manufacturing platforms because it can be easily deployed (by placing the mobile robots in the dedicated area), scaled up and down (adding or removing robots), and re-configured (adding or changing the type of robots). Second, it is more robust than the centralized manufacturing systems because malfunctions of individual robots can be kept local and will not break down the entire platform. In addition, the print size is not limited by the size of the printer since the mobile robots can roam over the entire production floor. Also, the use of multiple robots allows multi-color and multi-material printing, the cooperation between multiple processes (e.g., inkjet and extrusion), and the integration of pre-manufactured components to bridge the gap between traditional manufacturing and 3D printing.

Although C3DP provides a wide range of benefits, the scheduling of multiple robots to achieve collision-free printing is quite challenging. None of the research works discussed in sections 2.3 and 2.4 address the scheduling problems in C3DP. The approaches presented in section 2.3 have been able to provide very good solutions for multi-robots applications in MRTA problems, however, these solution approaches are not quite enough to solve a continuous manufacturing problem using multiple robots, as their primary target area of application

is the completion of discrete sets of tasks using multiple robots. Such discrete tasks could include assembly operations (where sets of pre-determined operations are undertaken) or subtracting manufacturing (e.g., drilling holes). On the other hand, while the entire process of C3DP has been discretized into multiple steps, the task of printing individual chunks is still a continuous process as it requires continual deposition of material both in space and time until the chunk is printed. So, the previously unoccupied space will gradually be populated with newly deposited material as the new chunks are printed, and we cannot wait until a chunk is fully printed to check for any constraint violation. This is because a conflict could occur between the printing robots while printing the chunks in close proximity. In addition to this, printing robots could also collide with the material deposited by another printer nearby. Thus, while multiple robots are working simultaneously to print new chunks in a shared environment in proximity, it is highly likely that at least one of the two such aforementioned conflicts could occur. Such conflict is not taken into consideration in the existing algorithm. This is because, to use the existing algorithm, the C3DP problem has to be posed as a discrete problem. So, if we take printing chunks as discrete tasks, check for such conflict cannot be done while the printing of chunks is ongoing, and one has to either check for constraints violation before printing the chunk or wait until the chunk is printed to check for conflict. If the check is conducted before printing, no violation will be detected, and if checking is conducted after, printing would have already failed by the time such conflict is realized. Thus, the existing solutions presented for MRTA problems cannot guarantee a collision-free solution. A new framework is needed to solve cooperative multi-robot additive manufacturing problems, which takes into account the conflict between the robots as they begin to deposit new material for assigned chunks in previously unoccupied space and can

guarantee a collision-free solution.

Similarly, the problems in IPPS might look similar to the problem in C3DP at first glance, the machines in job shop problems are stationary, and product moves from one station to another, which results in constraints that are quite different from the multi-robot problems where robots are moving from one location to another and are cooperating in shared space. The mobile nature of the robots brings additional complexity as there is potential for collision between the active robots as well as the robots and the printed parts. The robotic application 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 [57]. 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.

3 A HEURISTIC SCALING STRATEGY FOR COOPERATIVE 3D PRINTING

3.1 Overview

In C3DP, a printing strategy is composed of two separate yet related stages: chunking and scheduling. The chunking stage includes dividing a digital model at large into chunks (or printing tasks in more general terms) so that each chunk can be printed by a single printing robot. The scheduling stage deals with the assignment of the divided chunks to individual robots and generates a print sequence for each robot to achieve collision-free printing. In our previous work, we demonstrated this process with a two-robot system [58]. Due to the inherent complexity of the printing process and the lack of an existing mathematical formulation of the printing strategy, the logical next step is to search for a working strategy for a larger number of robots based on simple heuristics instead of seeking an optimal printing strategy. Thus, in this chapter, we present a heuristic-based scaling strategy – Scalable Parallel Arrays of Robots for 3DP (SPAR3). As mentioned in chapter 1, the chunking and scheduling stages are interrelated and depend on one another, and thus, one needs to understand chunking to understand scheduling. Therefore, first, the sloped-surface chunking strategy, the chunking strategy used to demonstrate the scheduling, is introduced in this chapter. The sloped-surface chunking strategy results in chunks with sloped surfaces so that the new material from adjacent chunks can be deposited on the slopes of already printed chunks, eliminating the need for any post-processing gluing work. Following the chunking method, the detailed explanation of the SPAR3 strategy is presented along with the evaluation metrics to cal-

culate the total print time and geometric constraints to ensure the feasibility of the print strategy (check whether the printing strategy results in collision-free printing). Finally, the SPAR3 strategy is used to simulate the print of two large print objects, and the analysis associated with it is presented, followed by the concluding remarks related to the chapter. The chapter is organized as follows:

3.2 Summary of sloped-surface chunking strategy presents a brief overview of the sloped-surface chunking strategy.

3.3 Heuristic Strategy for C3DP presents a first working strategy for C3DP based on heuristics.

3.4 Evaluation Framework presents the time-evaluation metrics to compute the total print time using directed dependency tree (DDT).

3.5 Geometric Constraints presents details of the approach of validating any print strategy using geometric constraints.

3.6 Validation of SPAR3 Strategy presents the results of two case studies, including a rectangular bar and topological map of Arkansas, which were validated using the geometric constraints and evaluated using evaluation metrics.

3.4 Conclusion and Discussion outlines the concluding remarks and major lessons learned from the study.

3.2 Summary of Sloped-Surface Chunking Strategy

Given a printing object, a chunking strategy is used to divide the part into smaller chunks. In this chapter, we adopt the chunking strategy with a sloped interface because this strategy has been studied and has been proven to maintain sufficiently strong adhesion

between chunks for the FDM process [59], the details of which is presented in Chapter 7. In the sloped interface chunking, a part is divided using a plane, first, along with one of either X- or Y-axis and then the remaining axis, such that each of these chunks has either a positive or a negative slope on each side, as shown in Figure 3.1. For example, chunk 0 in Figure 3.1 has positive slopes, whereas chunk 3 has two negative slopes (slopes adjoining chunk 0 and chunk 4) and two positive slopes. The angle of the plane determines the slope of the chunks, i.e., a plane of 30^{o} angle creates chunks that have a slope of 30^{o} angle. Different chunking parameters have to be defined, such as center chunk dimension (the depth of center chunk (e.g., chunk row 1 in Figure 3.1), chunk width (the width of the chunk), number of chunks.

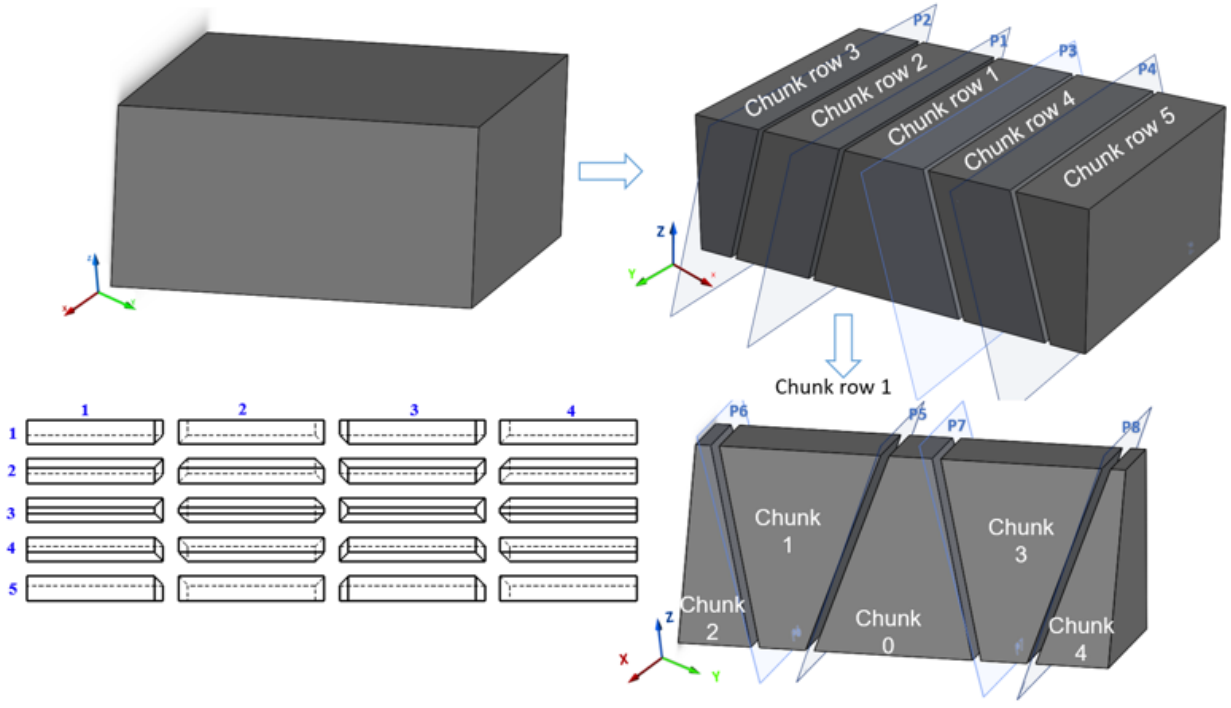

Figure 3.1: Chunking of a rectangular bar.

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 3.2(a) depicts

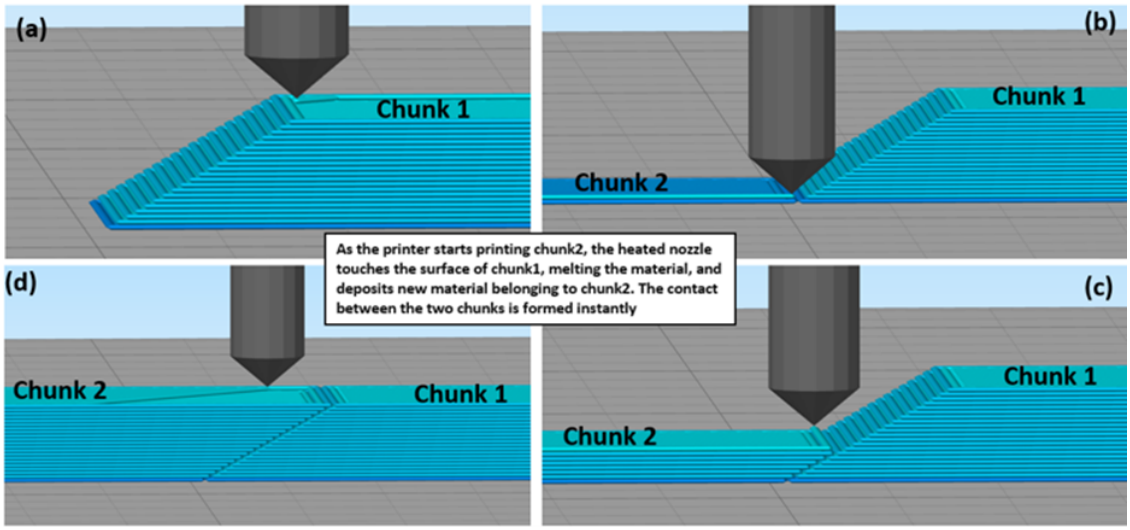

Figure 3.2: Process depicting how the connection between the chunks is formed.

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 chunk 1 (Figure 3.2(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 results in instant adhesion between the chunks. We have studied the mechanical strength of the part created using this chunking strategy [59], 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. A more detailed explanation of the mechanical strength and parameters associated with it are discussed in Chapter 7.

3.3 Heuristic Strategy for C3DP

The output of chunking, the chunks, along with the number of available robots, are taken as input for the generation of print schedule. Scheduling consists of two main aspects: chunk assignment and chunk scheduling.

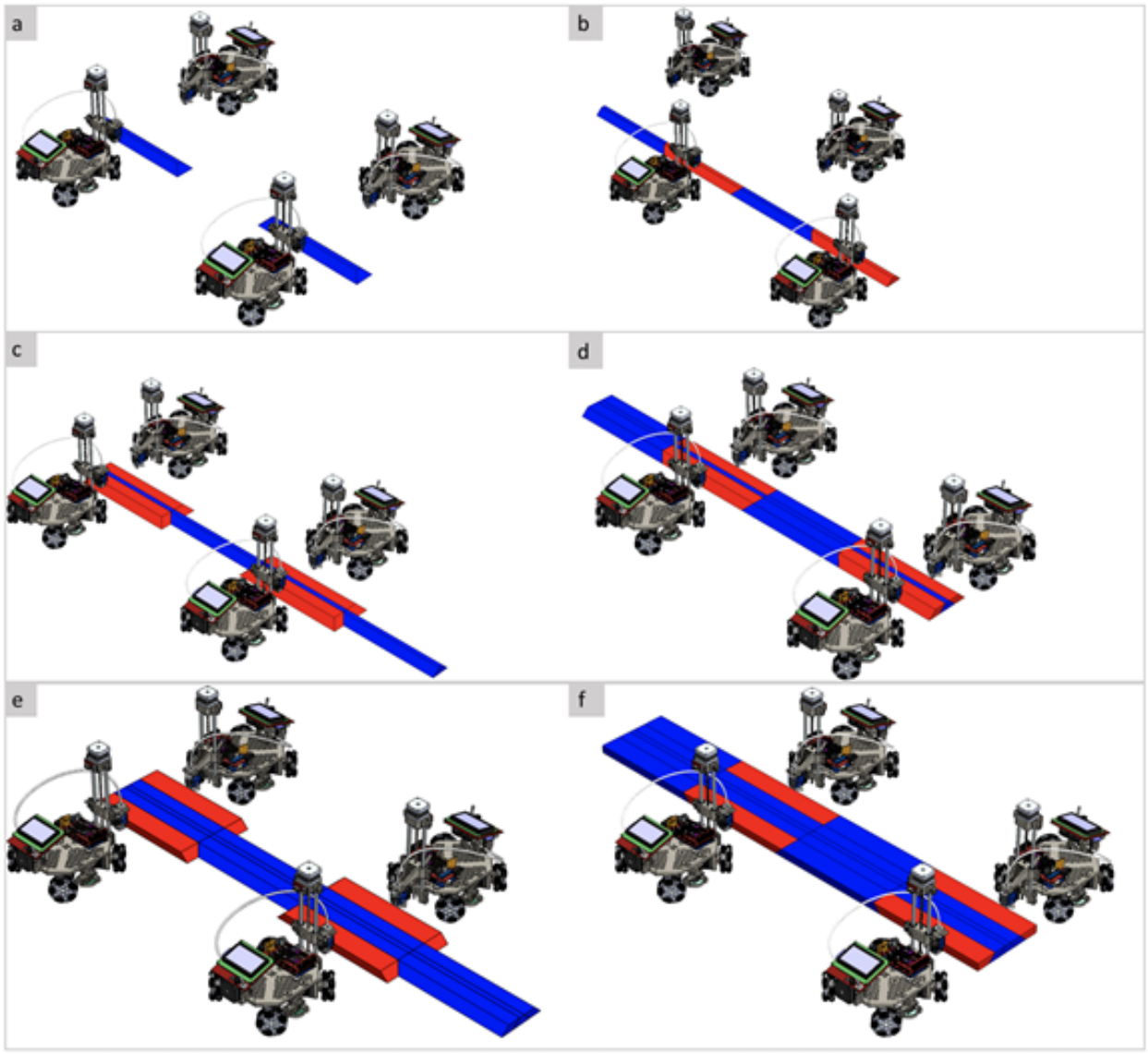

Figure 3.3: Illustration of the SPAR3 strategy with four robots.

- Chunk assignment: An example of a chunking outcome is depicted in Figure 3.1, which

has five rows and five columns of chunks. Each chunk is assigned to an individual robot. Each adjacent pair of chunks in a row is assigned to a single robot, such that there is a gap between the active robots (robots that are printing) at any given time to prevent collisions between them. For example, as illustrated in Figure 3.3 (a) and (b), chunks 31 and 32 (represented in Figure 3.1) are assigned to the same robot, and chunks 33 and 34 are assigned to the second robot. Doing so would prevent collision between the first and the second robot while they are working in parallel. Additionally, each row of chunks is assigned to only the robots that are on the same side of the center row so that there is no inter-row collision between them. For example, the row of chunks containing chunks 51, 52, 53, and 54 (represented in Figure 3.1) is assigned only to robots at the bottom in Figure 3.3.

- Chunk schedule: After the completion of chunk division and chunk assignment, a print sequence is generated based on the dependency relationship between chunks. Based on the dependency, the chunks can be divided into three types:
  - Seed Chunk: Seed chunks are the chunks that are printed first in a print job and have a positive bonding slope on all sides unless they are an end chunk. In Figure 3.1, chunks 33 and chunk 31 are the seed chunks.
  - Parent Chunk: Parent chunks are the chunks that need to be printed prior to printing any other chunks. In Figure 3.1, the seed chunks (chunk 33 and chunk 31) are the parent chunks of chunk 32 and chunk 34.
  - Daughter Chunk: Daughter chunks are those that cannot be printed until after their respective parent chunks are completed. The daughter chunk could either

be a gap chunk or dependent end chunk. In Figure 3.1, chunk 34 (dependent end chunk) is an example of a daughter chunk of the seed chunk 33. If a daughter chunk is located in between two parent chunks in the horizontal direction, we refer to it as a gap chunk. In Figure 3.1, chunk 32 is referred to as a gap chunk.

Using the printing object shown in Figure 3.1 as an example, the SPAR3 strategy in conjunction with the sloped surface chunking method is depicted in Figure 3.3. Chunks are assigned to four 3D printing robots, and printing begins at the center of the printing area and then expands into two opposing rows of robots. First, one row of robots prints every other chunk (seed chunks) as shown in Figure 3.3(a) while the others standby at the safe distance to avoid collision with the active robots. So, the robots printing the seed chunks will be the only group of robots that accomplish the center chunks. Once the seed chunks are complete, the same initial robots move over to print the gap chunks in order to fill the gap between the parent seed chunks (Figure 3.3(b)). After the completion of the central chunk row, the active robots retreat to begin printing the next row of chunks. Meanwhile, the robots on standby become active and begin printing the second row of chunks on the other side of the central row (Figure 3.3(c)). Both sets of the active robots follow the same strategy, i.e., print parent chunks →move over to fill the gap → print the daughter chunks → move to next row, until the print job is complete, as shown in the snapshots of Figure 3.3. The actual printing using the SPAR3 strategy, using two robots, is demonstrated in Figure 3.5. While the SPAR3 strategy can enable printing on either side of the center row, as highlighted above, the printing demonstration shows printing expansion taking place only on one side due to the lack of availability of more robots for printing during the time. Thus,

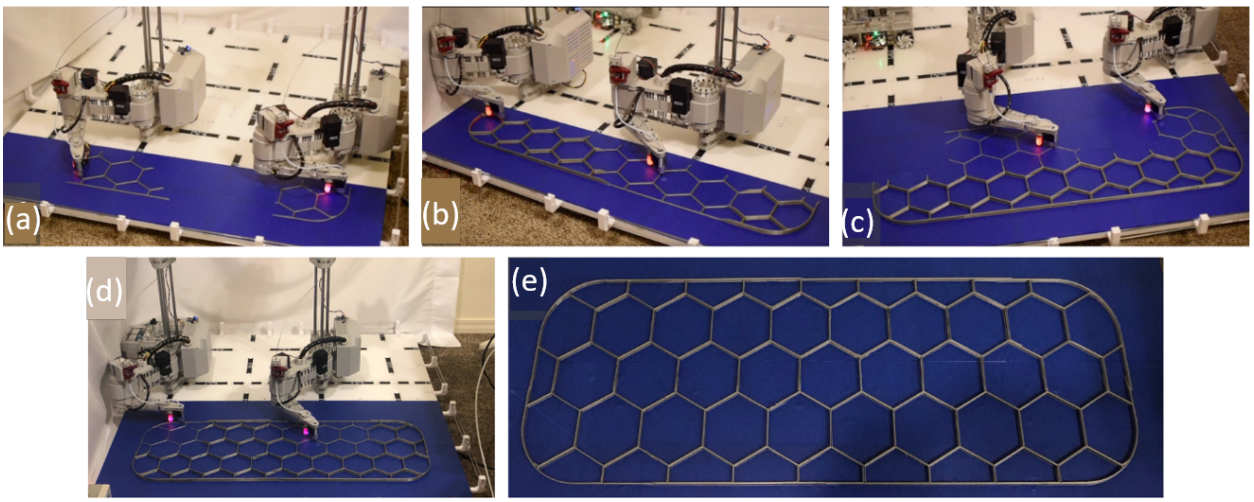

Figure 3.4: SPAR3 strategy implemented using two robots. Instead of expanding on both side of center row, it expands on one side only due to lack of availability of robots at the time of printing

the heuristic approach, SPAR3 strategy, used in conjunction with sloped-interface chunking strategy, can be outlined in the following manner. The first and the second steps are related to chunking, the third step is related to chunk assignment, while the fourth and the final step are related to chunk scheduling.

1. Determine the maximum number of chunk columns using equation (3.1) and a minimum number of chunk rows using equation (3.2). The details of equations 3.1, 3.2, and 3.3 are presented in chapter 7.

$$Max.\ number\ of\ chunks\ column\ (n_{cols}^{max}) = \frac{Length\ of\ the\ part}{Width\ of\ the\ robot} \tag{3.1}$$

$$Min.\ number\ of\ chunk\ rows\ (n_{rows}^{min}) = \frac{Width\ of\ the\ part}{Reach\ of\ printer} \tag{3.2}$$

2. Determine the number of chunks based on the number of robots available for printing

along with the values obtained from step 1 using equation (3.3). The total number of chunks) along with the total number of robots available for printing are the inputs for the subsequent step, i.e., chunk scheduling.

$$Total\ number\ of\ chunks = n_{cols} \times n_{rows} \tag{3.3}$$

3. The chunks assignment is done based on the proximity of chunks. For example, the robot that prints the center chunk of the center row is also assigned with the chunk next to it in order to minimize unnecessary movements. The alternate parent chunks are assigned to different robots. Their daughter chunks are assigned to the same robots such that once the parent chunks are printed, the robots can start working on the adjacent chunks (daughter chunks) without repositioning moves.

4. Chunk scheduling:

   (a) The printing begins with the center seed chunks with half of the total number of robots available.

   (b) Once complete, these robots then move to print the daughter chunks in the center row.

   (c) Once the center row is complete, the active robots move back to start working on the second row. The remaining robots become active and start working on parent chunks on the second row on the other side of the center row.

   (d) The printing of the daughter chunk follows afterward. This process continues until the part is complete.

3.4 Evaluation Framework

While it is possible to illustrate a printing strategy as presented in Section 3.3, the lack of a formal language to describe a printing strategy makes it difficult to evaluate the performance of the printing strategy. This section presents a graph-based language rooted in a directed dependency tree (DDT) to capture the most critical information of a printing strategy – the dependency relationship between chunks and the sequence of a printing process. Based on the DDT description, we develop a framework to estimate the total printing time of a given printing strategy. Based on the insights obtained from the SPAR3 strategy, we also formulate a set of constraints that a valid printing strategy must satisfy. If used in conjunction with DDT, these constraints can facilitate the development of new (and even optimal) printing strategies and meanwhile evaluating their validity and performance.

3.4.1 Chunk Dependencies and Directed Dependency Trees

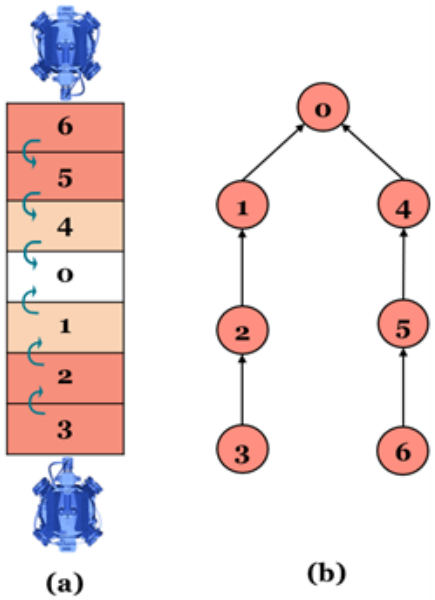

Figure 3.5: (a) Simple two-robot chunking (b) dependency tree.

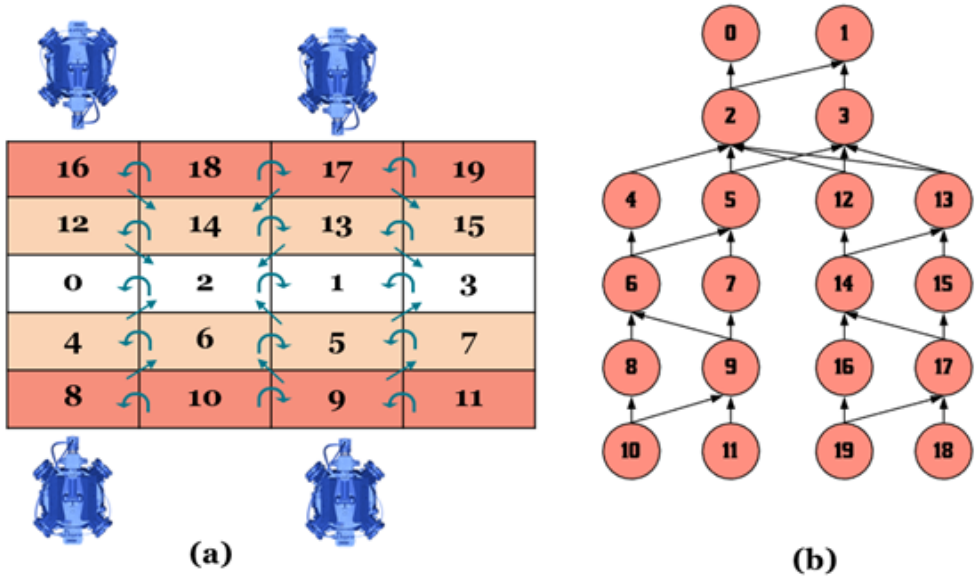

Figure 3.6: (a) Four-robot chunking (b) dependency tree.

One of the most important aspects of a valid printing strategy is to clarify the dependency relationship between chunks. A tree is a graphical representation that has been widely used to describe hierarchical relationships in various disciplines (e.g., computing, network representation). We adopt and adapt the tree concept, specifically the directed dependency tree (DDT) [60, 61], in our study to describe the dependency between chunks and printing sequences. Specifically, the chunks are represented as nodes, and the dependency relationships are represented as directed edges between nodes. The print sequence starts at the top of the tree and moves down along the edges of the tree. The chunks that are not dependent on one another can be printed by multiple robots in parallel. For example, Figure 3.5(a) shows a printing scenario where two robots work together on a workpiece that consists of 7 chunks (labeled 0 through 6), and Figure 3.5(b) shows the corresponding dependency tree. The top row with the chunk labeled 0 is printed first, followed by chunks 1 and 4, and so on

and so forth.

The SPAR3 strategy can be described in a similar fashion. Figure 3.6 shows the DDT of SPAR3 strategy when printing an object divided into 20 chunks (labeled 0-19) with four robots, which yields more intricate dependencies. The nodes show multiple dependencies compared to one dependency per node in Figure 3.5(b). In addition, there are cross-column dependencies. For example, node 2 has a dependency relation with node 0 of the same column as well as with node 1 of a different column. Figure 3.3 illustrates the actual printing scenario generated using the DDT in Figure 3.6(b).

Due to the rich information embedded in the dependency tree, many useful metrics from graph theory could be used to quantify and evaluate the performance of the printing strategy. For example, the depth of the tree (the number of rows in a tree) represents a total number of sequences (total number of printing steps). In contrast, the width of the tree (the total number of columns) determines the number of robots needed in printing. But after all, we are mostly interested in the total print time. To this end, we focus on two factors: a) the time that a robot takes to complete the current chunk and b) the time that a robot or group of robots take to complete its dependencies (i.e., the chunks that needs to be printed prior to printing the current chunk). In order to simplify the calculations, the time it takes for repositioning moves (the time it takes for the printer to move from the end of one chunk to the start of the next chunk) was neglected when implementing the SPAR3 strategy. The reasons for doing so are twofold:

1. While calculating the repositioning time, the path planning issue inevitably becomes a concern. For example, there could be printed materials on the way of a robot moving from point A to point B. Therefore, a collision-free path must be solved in order to

realize the printing schedule. This is especially true for strategies other than SPAR3 that require more movement on complex paths in between print cycles. The path planning issue for the multi-robot system in itself is an NP-hard problem. The integration of path planning into DDT for C3DP is our ongoing research which aims to develop a more comprehensive framework [62]. But it is out of the scope of this chapter.

2. For the SPAR3 strategy, the repositioning time does not result in significant errors in total print time because the chunks are adjacent to each other, and thus the robots do not have to travel much to get to the start of the next chunk. This is especially true for simple geometries, but as the complexity of the geometry increases, the repositioning time may have an impact on the total print time and will have to be taken into account. This can be observed in the case studies presented in the chapter. The error percentage between the calculated print time and the simulated time is larger for a map of Arkansas, which is geometrically more complex than a rectangular prism that has a smaller error percentage between the two printing times.

For a given DDT D, for node c_i, let $t(c_i)$ represent the amount of time a robot takes to print this single chunk. Then, the total time needed to complete printing the chunk c_i is the sum of the time it takes to complete printing all of its dependencies, $T[c_i^1, \ldots, c_i^{n_i}]$ and the time it takes to print this chunk, $t(c_i)$. This can be expressed as the following recursive function, where T outputs the completion time of a chunk at the node, c_i and $[c_i^1, \ldots, c_i^{n_i}]$ represents all dependencies of the chunk at that node. For the nodes that do not have dependencies (for e.g., chunks 0 and 1 in Figure 3.6(b)), 0 is taken as the maximum value in the following

equation.

$$T(c_i|D) = max[T(c_i^1|D), \ldots, T(c_i^{n_i}|D)], 0 + t(c_i), i = 1, 2, , 3, \ldots, N, \tag{3.4}$$

where, n_i is the number of dependent chunks of chunk c_i and N is the total number of chunks in a printing schedule. For example, for the DDT shown in Figure 3.6(b), the time at which chunk 5, which has chunks 3 and 2 as its dependencies, finishes printing is given by

$$T(c_5|D) = max[[T(c_5^2|D), T(c_5^3|D)], 0] + t(c_5)$$

Chunks 2 and 3 have chunks 0 and 1 as their dependencies, so the above equation can be further expanded to the following:

$$T(D, c_5) = max[(t_0 + t_2), (t_1 + t_2), (t_1 + t_3)] + t(c_5)$$

Using the same logic, equation (3.4) translates to the total time needed to print the entire sequence, T_{total}, in a dependency tree, D, with N chunks, the sum of the time it takes to print the last chunk, and the time it takes to print all of its dependencies as described in equation (3.5).

$$T_{total} = T(c_N|D) = max[T(c_N^1|D), \ldots, T(c_i^n|D)] + t(c_N) \tag{3.5}$$

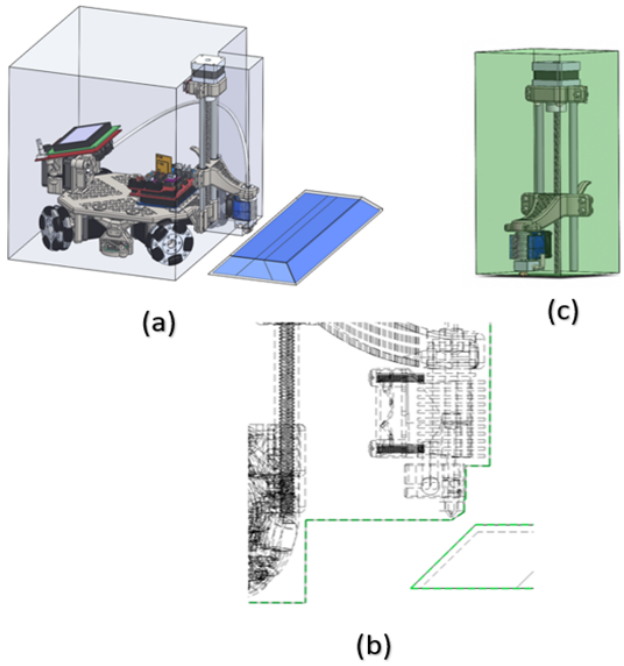

Figure 3.7: (a)AS of a robot and printed chunk (b) Side view (c)AS of robot reduced to z-stage.

3.5 Geometric Constraints

Once a printing strategy is described by a DDT, it becomes convenient to generate new printing strategies by using a tree generator to create new DDTs. However, not all DDT will be valid due to potential physical constraints. For example, to print the object shown in Figure 3.1, a chunk with a positive slope (e.g., chunk 1 in Figure 3.1) needs to be printed prior to printing adjacent chunks with a negative slope (chunks 2 and 3 in this case). Otherwise, there will be a collision between the nozzle of the printer and the printed chunk. Therefore, we need to formulate a set of constraints against which a printing strategy can be evaluated. These constraints can then serve as a sufficient condition to validate a printing strategy. In other words, if a printing strategy does not violate any of the constraints, the printing strategy should be valid, and the printing process will be guaranteed collision-free.

If there is no geometric constraint, any printing strategy will be valid, and all chunks can be printed simultaneously, which is unrealistic in real printing scenarios. Therefore, the only constraints that can make a printing strategy invalid are geometric constraints. To formulate a set of geometric constraints, the space occupied by printing robots as well as the printed chunks need to be defined. In doing so, we need to ensure that the geometric definition accurately represents the spatial constraints, and at the same time, these definitions are simple enough for efficient computation. Therefore, a balance between accuracy and computational efficiency must be achieved. In this chapter, we use the concept of the bounding box to define and formulate the geometric constraints:

1. Accessible space of a robot: We define accessible space (AS) of a robot, $AS_{R,i}$ as the 3D space occupied by the printing robot, i. We adopt a combination of 3D geometries such as cuboids to represent this space, as shown in Figure 3.7. The defined AS of a robot is dynamic as the position of the AS changes as the robot moves in the XY plane. Meanwhile, the shape of the AS may change if the nozzle height changes in the Z-direction. In order to make this approach more applicable to a robotic arm and SCARA 3D printers, only the z-stage of the print robot, as shown in Figure 3.7(c), can be used.

2. Occupied space of a printed chunks: We define occupied space of printed chunks,AS_c as the 3D space that is being occupied by the chunks that have already been printed, including the finished portion of chunks under printing. The shape of the occupied space is dynamic as the shape will change as the printing progresses. For example, upon the completion of the central seed chunk, the shape of occupied space will be the

same as the shape of that chunk, i.e., a trapezoidal prism.

3. Swept volume of a robot: Swept Volume, $SV_{R,i}$ is a volume formed by accessible space of a robot (AS) as robots move between the extreme points of chunks in XYZ space while printing a chunk. Since swept volume is a function of time, it can be defined at three different levels: chunk-level, layer-level, and line-level (i.e., the G-code level).

   (a) Swept volume at chunk level is the swept volume defined as the robot moves between extreme points of the chunk that is being printed in all three dimensions. It is defined over a longer period of time (from the start to the completion of a chunk).

   (b) Swept volume at layer-level is swept volume defined as a robot moves from one extreme point to another of a particular layer in XY-plane and is defined over a slightly shorter period of time (from the start to completion of a layer).

   (c) The line-level swept volume is defined as the robot moves from the beginning to the end of the G-code line command. This is defined over the shortest time period among all three (from the start to the end of the G-code line command in slicer).

With these concepts defined above, we develop geometric constraints. For any valid scaling strategy for C3DP, the following two conditions must be satisfied:

1. A robot, i, does not collide with any other robots if and only if the swept volume of the robot, $SV_{R,i}$, does not overlap with a swept volume of other robots, $SV_{R,j}$, and $i \neq j$ at any time during the entire printing process, i.e.,

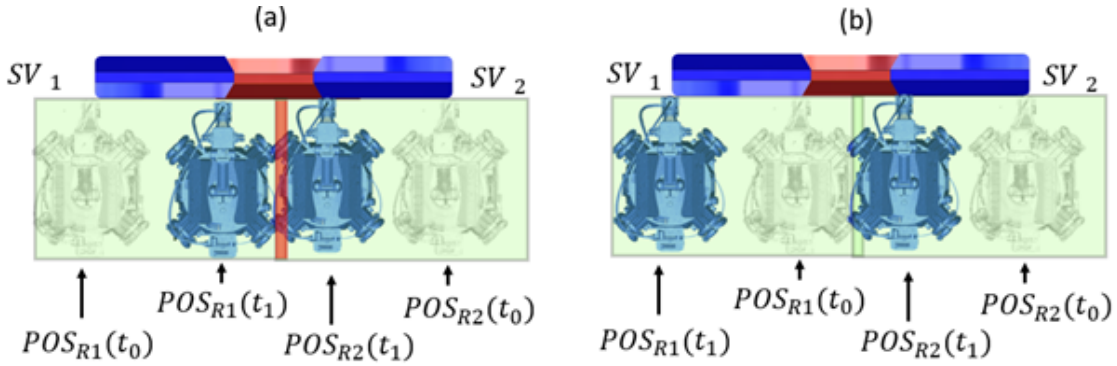

Figure 3.8: Two robots have overlapping SV but the robots (a) collide (b) Do not collide

$$SV_{R,i}(t) \cup SV_{R,i}(t) = \phi, i = 1, 2, 3 \ldots, n; j = 1, 2, 3 \ldots, n; ji \tag{3.6}$$

2. A robot, i, does not collide with already printed chunks if and only if the accessible space of a robot ($AS_{R,i}$) does not intersect with the occupied space of printed chunks (AS_c).

$$AS_{R,i}(t) \cup AS_c(t) = \phi, i = 1, 2, 3 \ldots, n \tag{3.7}$$

Equations (3.6) and (3.7) provides the mathematical representation of the geometric constraints that can be algorithmically checked for a given DDT of a printing strategy. Equation (3.6) will be first checked on the chunk level. No further check is needed if there is no violation on the chunk level. On the other hand, if there is a violation of equation (3.6) at chunk level, there is potential for collision between the printing robots, i.e., there is an intersection between the volumes swept out by the active robots while completing the assigned chunks, as marked in red in Figure 3.8(a). This, however, does not assure collision between the robots, as shown in Figure 3.8(b). For the robots to have a collision, both of them have to occupy the same location at the same time, as shown in Figure 3.8(a). In order

to avoid such scenarios (scenario shown in Figure 3.8(b), where the swept volumes of the robots intersect each other, but the actual collision does not occur), we do the check at the layer level. No further check is needed if the layer-level check passes. Otherwise, a line-level check will be conducted for a reason specified earlier. If the line-level check fails, we can conclude the strategy is invalid. The reason for defining the swept volume on three different levels is because it is most computationally efficient to do a chunk-level check, but it is also the most conservative since many valid strategies can be ruled out if the only chunk-level check is performed. The layer-level and line-level checks are more accurate but require more frequent checks, which is computationally taxing. Thus, for any printing strategy represented by a DDT, along with the sliced G-code for each chunk, we can perform a check against the geometric constraints in equations (3.6) and (3.7) based on the timing sequence defined by the DDT. This provides a simple, go-or-no-go type of output to ensure the validity of any printing strategy represented by a DDT.

3.6 Validation of SPAR3 Strategy

Before implementing a printing strategy, it needs to be ensured that the strategy is valid. In order to do so, geometric constraints developed in Section 3.5 are to be scrutinized against the generated print sequence to make sure that no constraints are violated during the print sequence. In this section, we follow the steps outlined below to validate the proposed SPAR3 strategy, which can also be used to check the validity of any other printing strategies:

1. Chunking: Generate chunks for a given CAD model using the Chunker.

2. DDT Construction: Construct the DDT for the given printing strategy (e.g., SPAR3).

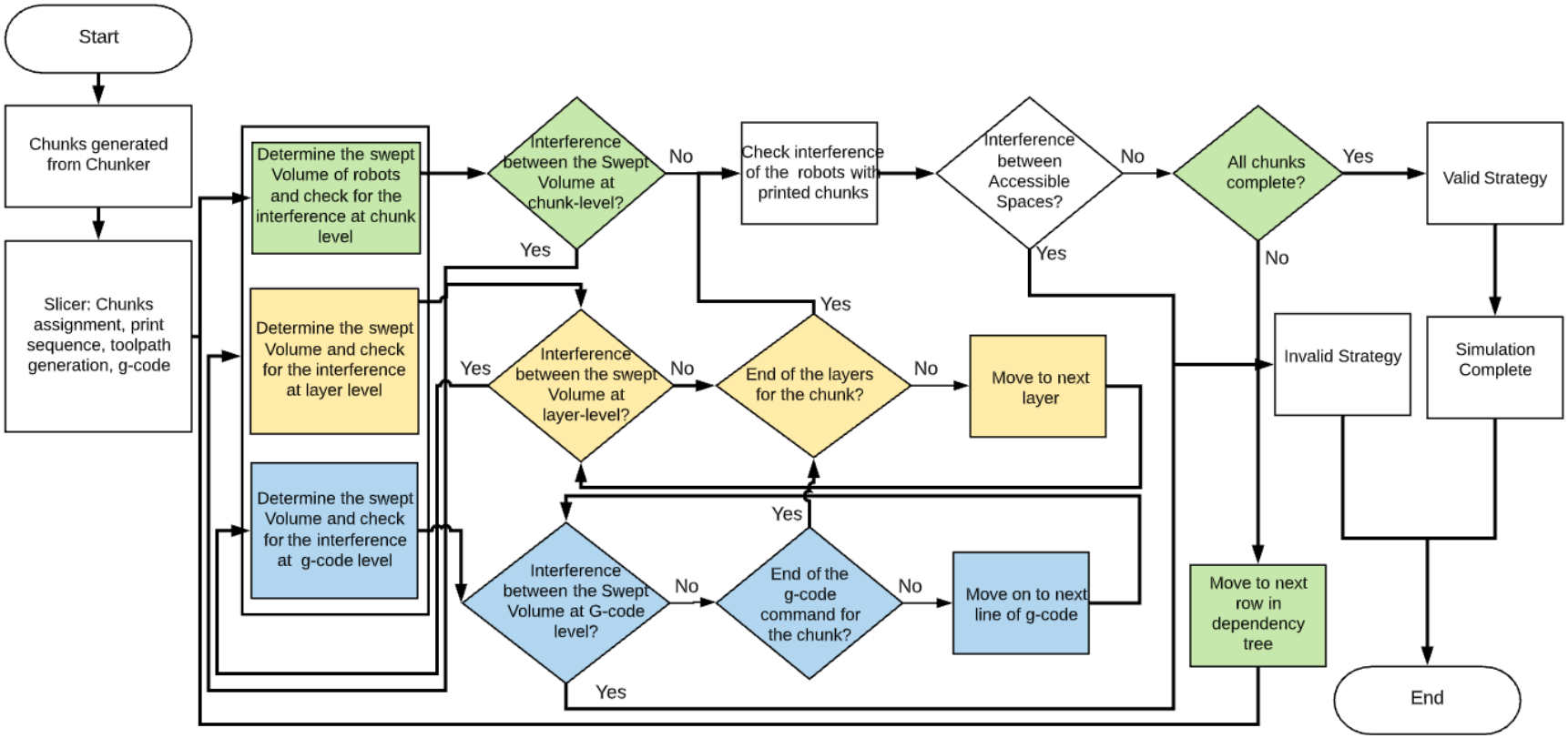

Figure 3.9: Validation flowchart of SPAR3 strategy

Table 3.1: Parameters settings for the simulation

Robot width:	16cm
Robot build depth:	4cm
Robot printhead slope:	60°
Slice thickness:	1.6mm
Infill type:	Solid
Time per frame:	140s

3. Slicing: Generate toolpath (e.g., G-code) for all the chunks.

4. Constraints checking:

   (a) Check the constraints in equation (3.6)

      i. Chunk-level checking: if pass, go to step 4.b; otherwise, go to step 4.a.II.

      ii. Layer-level checking: if pass, go to step 4.b; otherwise, go to step 4.a.III.

      iii. Line-level checking: if pass, go to step 4.b; otherwise, go to step 5.

   (b) Check the constraints in equation (3.7)

5. Output: if any check in step 4.a or 4.b fails, the printing strategy is invalid; otherwise, it is a valid strategy.

The flow chart of this algorithm is shown in Figure 3.9. An algorithm is developed to check the validity of the SPAR3 strategy, and the results show it is valid.

3.6.1 Simulation Results and Discussion

The SPAR3 strategy and the evaluation framework were tested on two different jobs with different complexity of geometry: a simple rectangular prism and a topographical map of the State of Arkansas. Based on the SPAR3 strategy, DDT was generated for each model. The validity of the generated sequence from the DDT was checked by ensuring

none of the defined geometric constraints were violated. Once verified, the strategy was implemented using our previously developed chunk-based slicer [58], from which the G-code file is generated, and the printing process can be therefore simulated. In this paper, our previously developed simulator [58] was extended for multi-robot simulation. The results of printing time obtained from both the theoretical estimate (i.e., equation (3.5)) and the computer simulation are compared and presented in Table 3.2.

In [58], we developed a chunker (that takes a CAD model and divides it into chunks based on set criteria), a slicer (that slices those chunks into layers and generates G-code commands of printing, such as tool path, speed, material extrusion, etc.), and a simulator (that visualizes and animates the printing process based on those commands) for the two-robot printing strategy. In this chapter, we expand the functionality of the chunker, slicer, and simulator software to allow an arbitrary number of robots.

The simulator is built in a Blender environment using a Python script and reads the text commands (G-code commands) which are generated from our chunk-based slicer and then animates the motions based on the commands. It takes the same G-code command that an actual 3D printing robot does and uses the same time prediction algorithm that is used in conventional 3D printers. The repositioning moves of mobile robots are included in the G-code and thus accounted for in the time estimation. While there is some discrepancy between the actual printing time and the estimated print time [63] due to accelerations and decelerations settings of the printer, the simulation time is fairly close to the actual print time (although the discrepancy can increase with the complexity of the geometry). To reduce the computational cost, simulations are rendered such that a single frame represents 140 seconds of real-time. This reduces the time taken to render a scene by reducing the total

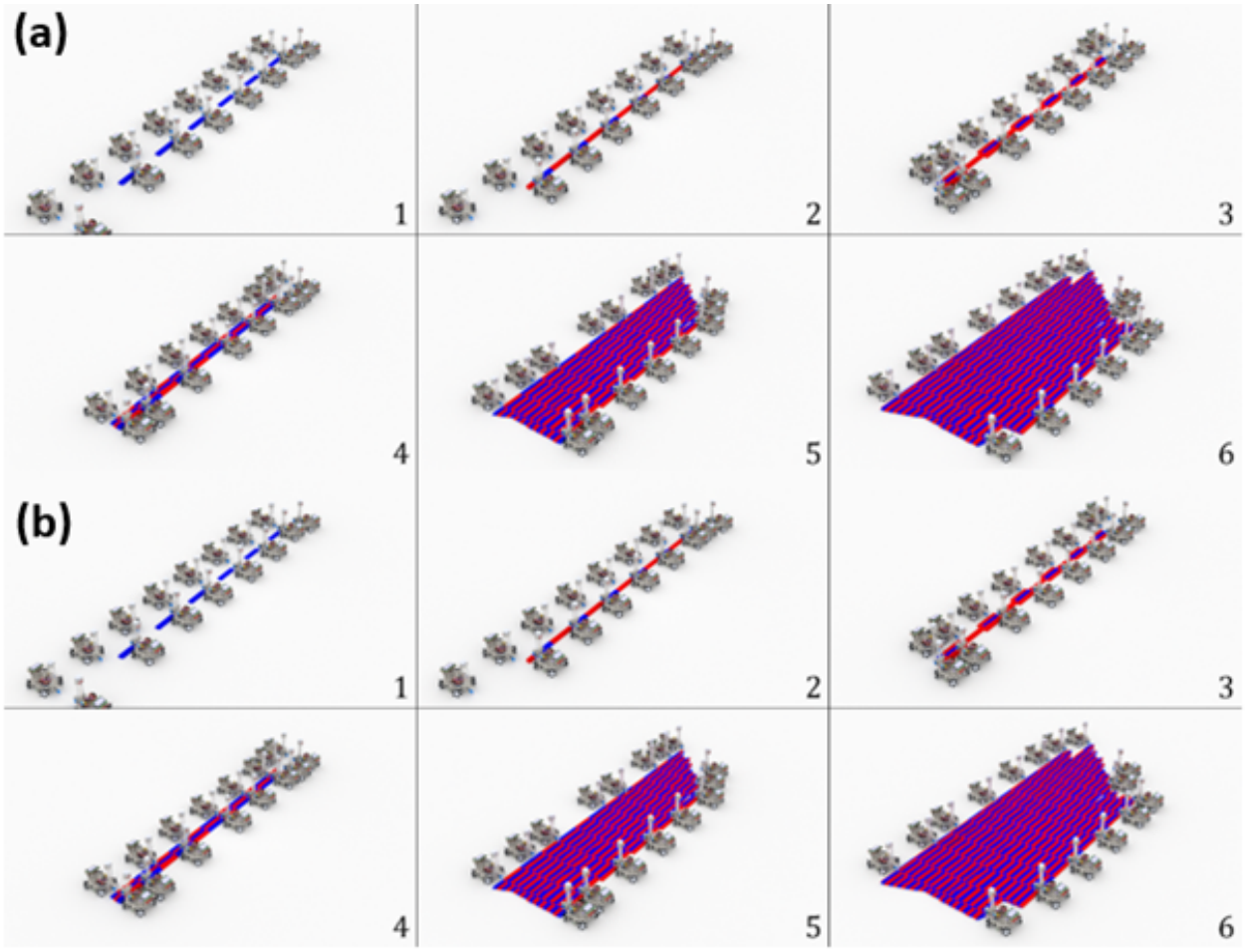

Figure 3.10: The printing of (a) a rectangular prism (b) an Arkansas topographic map.

number of frames in the simulation but at the same time gives us accurate print time. Our testing methodology is to compare the number of frames needed to print a 3D object fully, i.e., the amount of real-time it would take to complete a print – across three strategies: (1) Single robot printing, (2) Two robot printing, and (3) the SPAR3 strategy with up to 16 robots. This simulated data will be compared against the predictions from our DDT-based evaluation detailed in Section 3.4.

Table 3.1 shows the default parameters for our simulation environment. The first model is a simple rectangular prism, $280cm \times 24cm \times 2cm$ with a total volume of $13,400cm^3$.

Table 3.2: Estimated time vs simulated time for two models

	Rectangular Prism Model			Arkansas Model		
Robots	Estimated Time (h)	Simulated Time (h)	Speedup	Estimated Time (h)	Simulated Time (h)	Speedup
1	188.46	190.63	N/A	257.02	258.00	N/A
2	120.09	120.09	1.59	162.24	162.21	1.59
4	61.41	60.98	3.13	101.81	101.7	2.54
6	41.49	41.34	4.61	74.9	72.99	3.53
8	31.34	341.31	6.09	60.16	58.14	4.44
10	25.51	25.39	7.51	49.89	47.76	5.4
12	21.39	21.39	8.91	42.93	40.13	6.43
14	18.51	18.51	10.03	37.88	36.59	7.05
16	16.57	16.53	11.53	37.68	35.58	7.25

The snapshots of the print sequence are depicted in Figure 3.10 (a), numbered 1 through 6. The second model is a topographic map of the State of Arkansas, approximately $232cm \times 87cm \times 2.5cm$ with a total volume of approximately $19,524cm^3$. The print sequence of this model is illustrated in Figure 3.10 (b) and is numbered 1 through 6 as well.

Table 3.2 shows the estimated and simulated time as the number of hours it takes for both printing jobs. The estimated time is calculated based on equation 3.5, whereas the simulated time is the total time it took the robots to complete the print job in the simulation.

Table 3.2 provides two very important pieces of information. First, the results indicate that the SPAR3 significantly speeds up the printing process. For example, if 6 robots are used to print a rectangular prism instead of 1, the print time shortens from almost 191 hours (nearly 8 days) to a little over 41 hours (less than 2 days). With only two robots, we see speedup results similar to the estimates from our previous work [58]. However, with the new SPAR3 scaling strategy, we are able to parallelize the workload to at least 16 robots (and potentially more for larger print objects). The speedup shows linear growth with the number of robots for a rectangular prism, as shown in Figure 3.11. The same graph also shows linear

growth at the beginning for the Arkansas model, but the growth is not as significant towards the end. We expect the rectangular prism model to follow a similar trend as the number of print robots employed increases further. This is expected because once the number of robots to fully parallelize the printing is achieved, adding a number of the robots would not result in a reduction of print time further as the additional robots are not utilized for printing. The upper limit (number of robots) at which the speedup stops improving can be obtained using modified Amdahl's law from our previous work [58].

The rectangular prism works very well with the SPAR3 strategy due to the fact that all chunks have roughly the same volume, and every robot has exactly two chunks to print per row. This maximizes the amount of parallelization that can occur at any given point in time. In comparison, the Arkansas topographical map has an irregular shape that results in the volume of chunks being significantly different. This causes multiple robots to wait for long periods of time before they can begin while others are still working on their assigned chunks because of chunk dependencies. As a result, the Arkansas model sees worse speedup (the orange dot line in Figure 3.11 compared to that of the rectangular prism shape (the blue dot line in Figure 3.11. This leads us to the conclusion that if we desire to further decrease the total print time using the SPAR3 strategy, the more uniform volumetric size of chunks is preferred. Second, it provides a comparison between the estimated time and simulated time. Figure 3.11 provides the graphical representation of the error percentage calculated using the estimated time and the simulated time against the total number of robots used. The estimated time for the rectangular prism shows a lower error percentage ($> 1.15\%$) and stays relatively steady with the change in the number of robots. On the other hand, the error percentage for the Arkansas model fluctuates (between $0.02 - 7.0\%$) and

is higher for a larger number of robots. This discrepancy is the result of the non-uniform volume of chunks. Additionally, unlike for the rectangular prism, the trend of error is not very obvious. Although the overall error is not very high, the error percentage fluctuates between different printing scenarios in both cases. In addition to the impact of non-uniform chunk volume, the repositioning moves are the most likely factor causing this fluctuation. Different printing scenarios with a different number of robots result in a different number of repositioning moves during printing, which causes the said fluctuation between different printing scenarios. Nonetheless, equation (3.5) provides a fast approach to estimate the printing time with reasonable accuracy.

3.7 Conclusion and Discussion

In this chapter, we demonstrated the use of a scalable parallel array of robots for 3D printing (SPAR3), which was based on human cognition. This is the first working strategy that enables multi-robot 3D printing and hence, does not guarantee optimality. The approach was demonstrated using two print simulations, a rectangular prism, and an Arkansas topographic map. In summary, this chapter makes the following contributions:

- We presented a heuristic scaling strategy that enables cooperative 3D printing with many robots.
- We constructed a formal mathematical language to describe a printing strategy using a directed dependency tree, which allows us to formulate a generic evaluation framework for different printing strategies. The evaluation framework can be used to check the validity and estimate the total printing time of any printing strategy. This method

provides a better estimate over the previously used Amdahl's law to calculate the total print time.

- In order to check the validity of a printing strategy, we developed a set of geometric constraints. A printing strategy is guaranteed to be valid if none of the geometric constraints are violated. These geometric constraints were used to check for robot-robot collision and robot-to-printed part collision.
  - To ensure no robot-to-robot collision occurs, swept volume of printing robots over a varied period is defined: chunk-level (beginning to the end of chunk printing), layer-level (beginning to the end of layer printing), and line-level (beginning to the end of G-code line command printing) and checked for a nonzero intersection.
  - To ensure no robot-to-printed part collision occurs, the accessible space of the printing robot and the occupied space of the printed part are defined and checked for a nonzero intersection.
- The SPAR3 strategy was then validated and evaluated using the developed framework. The evaluation framework and a simulation tool are used to estimate the total printing time of the SPAR3 strategy. Results show significant speedup as the number of robots increases.

To obtain an optimal printing strategy for cooperative 3D printing, it is paramount to have a solid framework to aid the development of one. And the framework developed using a heuristic search in this chapter provides just that. Thus, this chapter provides a solid foundation for the development and optimization of printing strategies in more complicated

situations, such as printing complicated geometric shapes with intricate chunk dependencies. The issue of scalability, which has been the Achilles' heel of AM as well as a difficult task for cooperative 3D printing (due to difficulties associated with logistics, task management, and collision avoidance), can be addressed with the SPAR3 strategy proposed in this chapter. Combined with the mathematical representation of geometric constraints as well as the evaluation metrics developed, it could be adopted to implement fully autonomous digital factories and be used to optimize the operation of the factories. In the following chapters, the foundation built in this chapter (evaluation framework, geometric constraints) is used to generate print schedules automatically, without relying on human heuristics.

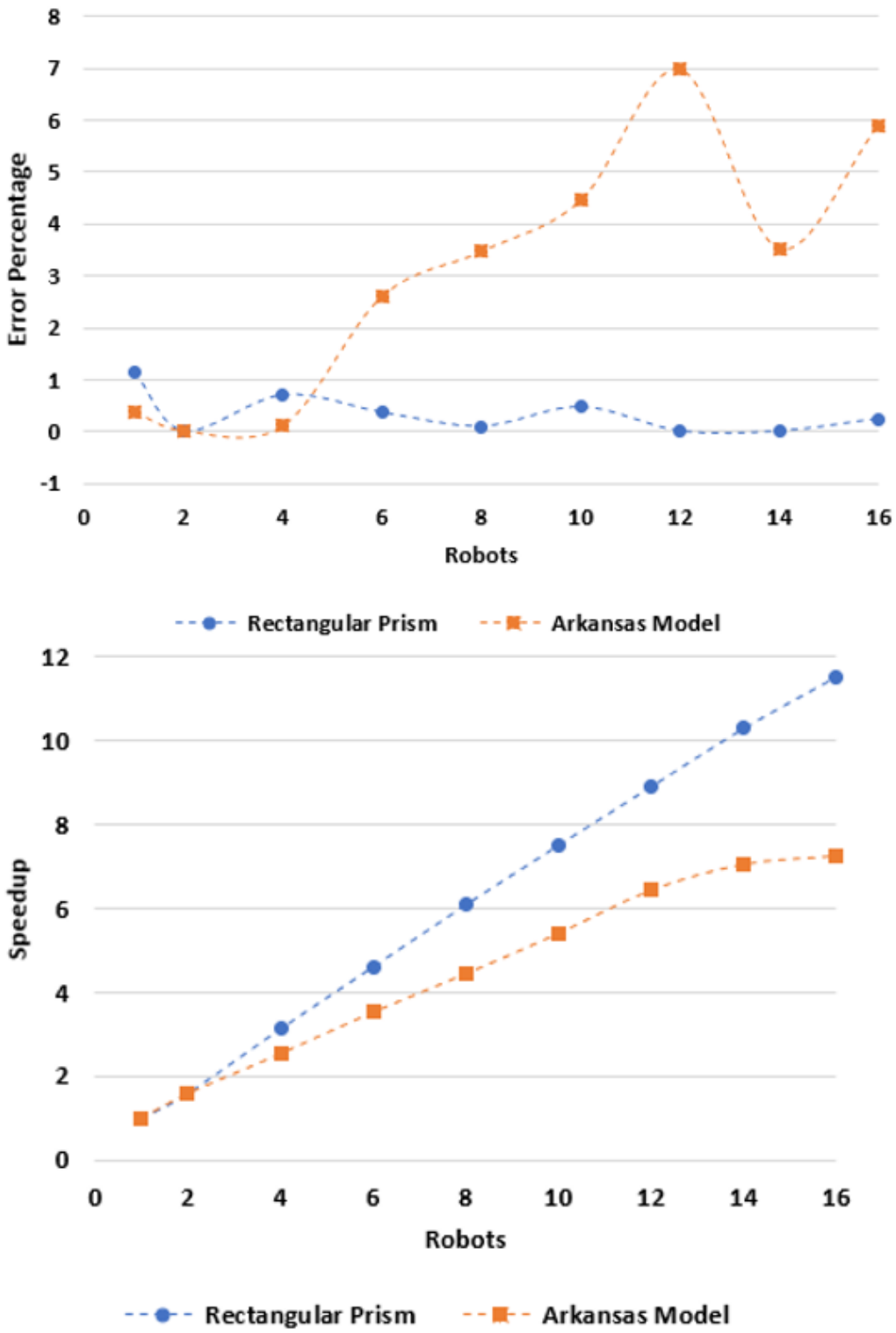

Figure 3.11: Graph of number of robots vs. the speedup (top) and number of robots vs. the error percentage (bottom).

4 A GENERATIVE FRAMEWORK FOR MULTI-ROBOT COOPERATIVE 3D SCHEDULING

4.1 Overview

In the previous chapter, we presented the first working scaling strategy for 3D printing using multiple robots. Still, the print schedules were based on human heuristics, and even though it is generalizable enough to be applied to different print scenarios, it lacks the sophistication to generate an efficient solution for a complicated geometry. If we only rely on human cognition, we might only be able to generate a handful of valid print schedules. Thus, to address limitations, we present an automatic generative framework for schedule generation in this chapter. The generative framework generates random print schedules using the concept of an adjacency matrix. The concept of the adjacency matrix is widely used in network science to demonstrate the existence of an edge between the two nodes. Once the print schedules are generated, they are evaluated and validated to ensure that the schedules do not result in a collision during printing.

At first glance, the problem addressed in this chapter might seem similar to the problems addressed in related research fields such as multi-robot research and Integrated process planning and schedule; however, it is not the case as outlined in chapter two. 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 [64], where the author highlights the need for 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 chapter 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.

The chapter is organized as follows:

4.2 Generative Framework for Automatic Schedule Generation introduces and describes the algorithm in more detail, including the detailed explanation of individual steps.

4.3 Case Studies details the two case studies used to demonstrate the algorithm.

4.4. Result and Discussion discusses the results and their implications

4.5 Conclusion outlines the concluding remarks and major lessons learned from the study.

4.2 Generative Framework for Automatic Schedule Generation

While a print schedule based on the simple heuristic, presented in chapter 3, 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. 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 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 section, a new

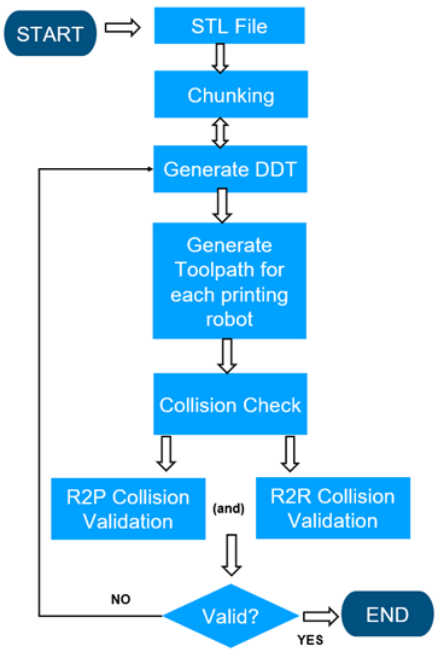

Figure 4.1: Flowchart of proposed generative framework.

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 is presented. 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 (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. Human heuristics were used to concatenate this information with the geometric dependencies to generate a full DDT in Chapter 3. Due to the limitation of human cognition, human heuristics might

not be able to explore every possibility. 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 in which human heuristics might not be capable.

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 4.1.

4.2.1 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) [65, 66] 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 4.2. 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 4.2, 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 the 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 [67] 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.

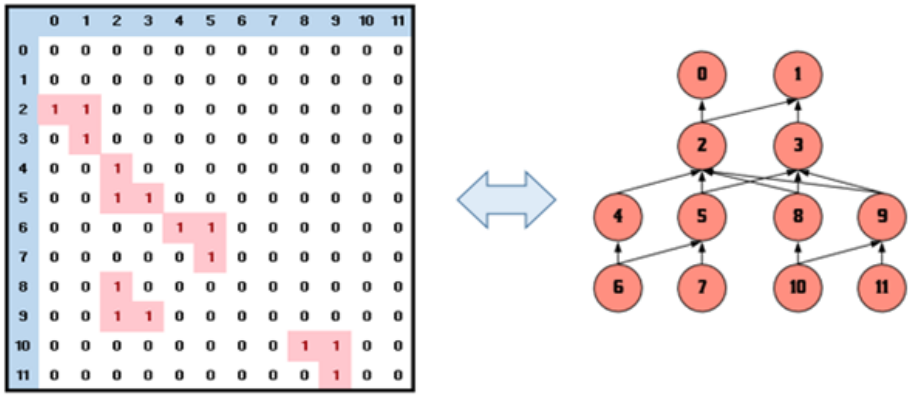

Figure 4.2: Adjacency matrix and the equivalent DDT.

4.2.2 Validation Using Geometric Constraints

Once the print schedules are generated, the validity of the print schedules is checked using the geometric constraints identified in Chapter 3. In that chapter, we identified geometric constraints to check the validity of the print schedules to ensure that the generated print schedule results in collision-free printing. 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 Chapter 3:

1. A robot i does not collide with already printed chunks. The mathematical formulation of these constraints was presented in the form of Equation (4.1) 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).

$$AS_{R,i}(t) \cup AS_c(t) = \phi, i = 1, 2, 3 \ldots, n, \tag{4.1}$$

where $AS_{R,i}(t)$ is the accessible space of the robot i at time t and, $AS_c(t)$ 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 (4.2) using the concept of swept volume of the printing robot (3D space occupied by the robot as it prints the entire chunk).

$$SV_{R,i}(t) \cup SV_{R,i}(t) = \phi, i = 1, 2, 3 \ldots, n; j = 1, 2, 3 \ldots, n; j \neq i, \tag{4.2}$$

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 a detailed description of the identified geometric constraints, can be found in Chapter 3. Thus, any generated print strategies can be validated using the geometric constraints presented in equations (4.1) and (4.2) to ensure that the collision does not take place between the printing robots as well as between the printed parts and printing robots.

4.2.3 Time Evaluation Using DDT

In addition to the generation of geometric constraints, we presented the idea of using DDT to calculate the total print time of a printing schedule in Chapter 3. 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 4.2 depicts a print schedule with four printing sequences and utilizes a maximum of four robots

(but only two are needed in two initial sequences. In the study, we presented equation (4.3) 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.

$$T_{total} = max(T(D, c_m))|\ m\ \epsilon\ [0, n-1] \tag{4.3}$$

4.3 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 chunk immediately. Ignoring the travel time between the chunks would not result in significant difference due to a) out of total time, the 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

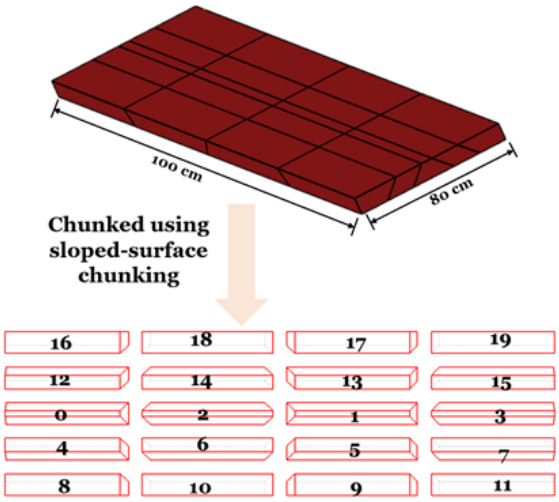

Figure 4.3: Rectangular block showing the chunks line, exploded view of the chunks.

significant discrepancy in calculating the total print time.

4.3.1 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 $100cm \times 80cm \times 1.5cm$ and have a total volume of $12,000cm^3$. 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 4.3. 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 4.3. If a 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 $16mm^3/s$ 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 an 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 4.3. The algorithm then generates an adjacency matrix that meets all the predefined criteria specified in Section 4.2.1. 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 4.2.2.

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 4.4. 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. This constraint is generated by considering only the z-stage of the print robot, shown in Figure 4.4(a). 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 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

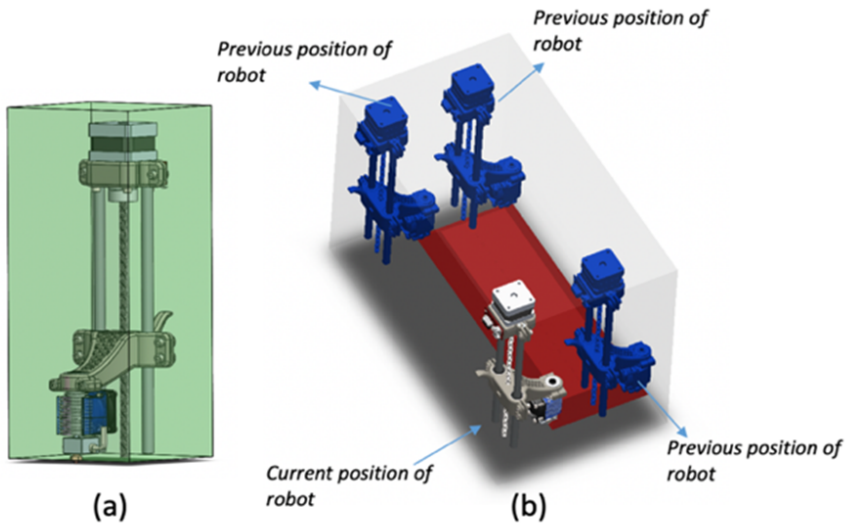

Figure 4.4: (a) AS of robot (reduced to z-stage) (b) SV of robot while print a chunk.

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 the generation and constraints check, the valid print schedules were evaluated using time metrics presented in Chapter 3. Each chunk takes about 10.42 *hours* to print, which is the maximum time it takes to complete each sequence.

4.3.2 Case Study II: Folding SUV Model

For the second case study, we use a toy folding SUV with a dimension of $157.6cm \times 140cm \times 3.6cm$. 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 more complex geometry, and as a result of this, the chunking results in chunks with variable sizes, volume, and shapes. The part and the resulting chunks obtained after chunking are presented in Figure 4.5. Since the part is larger than the first case study, to reduce the total print time, $40mm^3/s$ 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 4.6. This is a conservative approach and might eliminate some valid print schedules during collision check, but it drastically reduces the 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 the 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 4.6(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

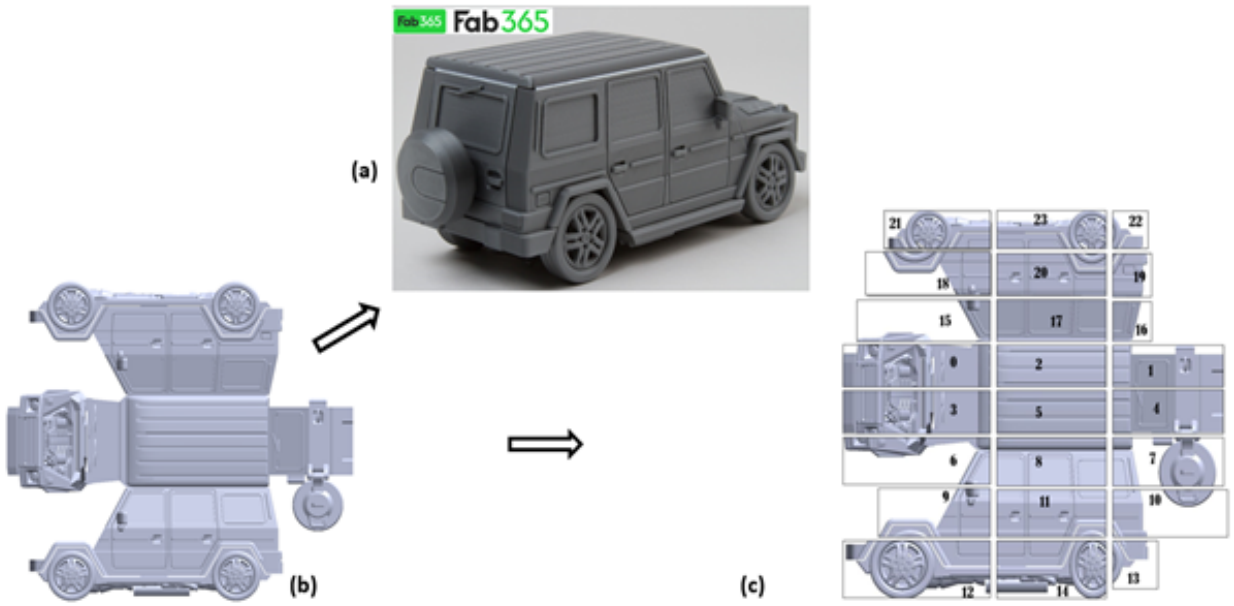

Figure 4.5: (a)3D model of a SUV vehicle (b) Unfolded STL model (c) Resulting chunks of the model.

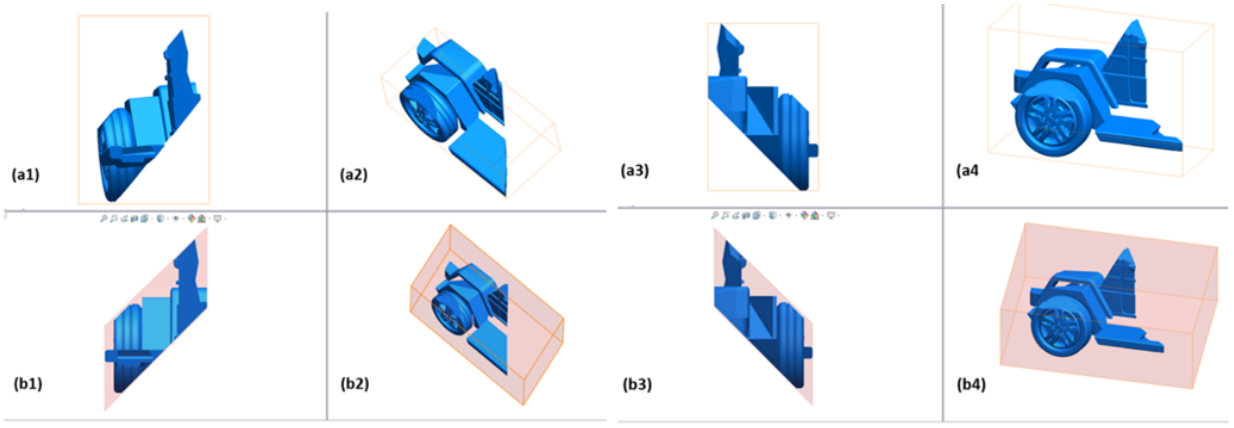

Figure 4.6: Actual shape of a chunk from Figure 7(c) (a1) Front View (a2) Isometric view (a3) Dimetric view (a4) Back view. MEV (b1) Front View (b2) Isometric.

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.

4.4 Results and Discussion

4.4.1 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 4.1. The print sequence in Table 4.1 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 a 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. The top five generated print schedules are presented in DDT format in Figure 4.7, and the print sequence developed using a heuristic approach is

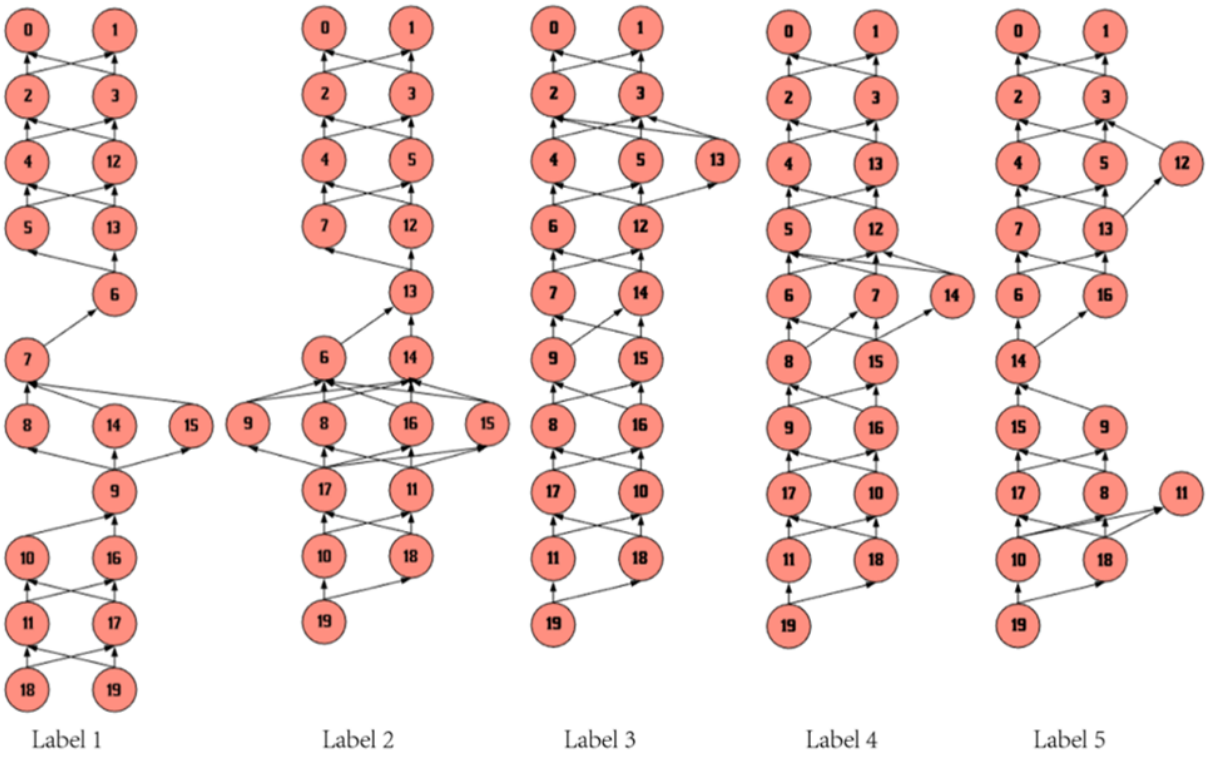

Figure 4.7: The DDT associated with the print sequence in Table 4.1.

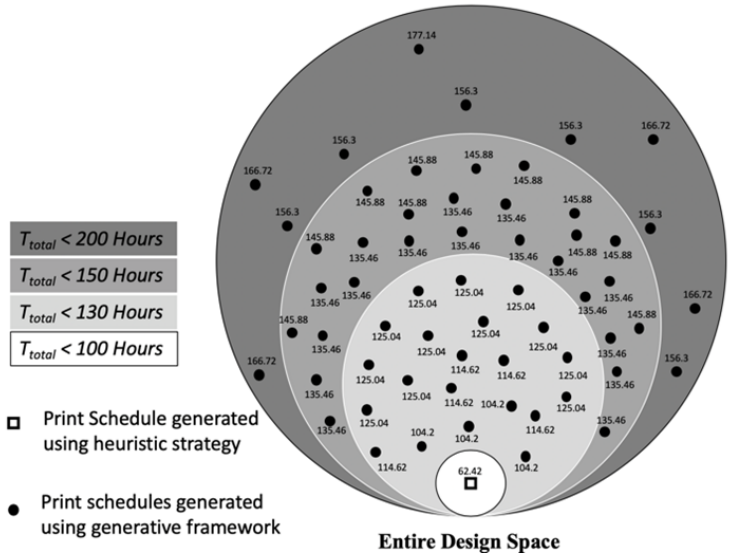

Figure 4.8: The plot showing the print time for each valid generated tree.

Table 4.1: Top five generated print schedule along with the heuristic strategy

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

presented in Figure 4.9(a). 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 schedule 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 smaller circles. The plot is presented in Figure 4.8. The total print time of the heuristic approach SPAR3 is also plotted in the graph for comparison, as marked by the square marker. As can be seen in the figure, the heuristic print schedule has a shorter print time, but the generative approach was able to generate diverse print schedules.

4.4.2 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 the case study I. The DDT representing the heuristic strategy SPAR3 is presented in Figure 4.9(b). It can be observed that during the majority of print sequences (i.e., different layers of the DDT), only half

Table 4.2: Top five generated print schedule along with the heuristic strategy

Label	Print Sequence {sequence number: chunk numbers}	Time (hrs)
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 number of available robots is utilized compared to the 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 4.2. The same five print schedules highlighted in Table 4.2 are presented in DDT format in Figure 4.10. As expected, the generator resulted in print schedules with much fewer parallel print sequences.

The total print time for the print schedules generated using the algorithm is much longer ($\sim$ 1.67 times longer for case study I and $\sim$1.52 times longer for case study II)

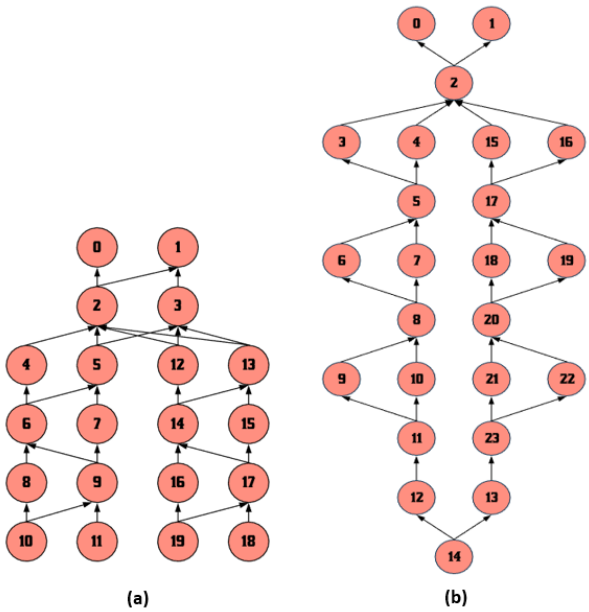

Figure 4.9: The DDT of heuristic strategy for (a) Case Study I (b) Case study II.

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 nonuniform, 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 is 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

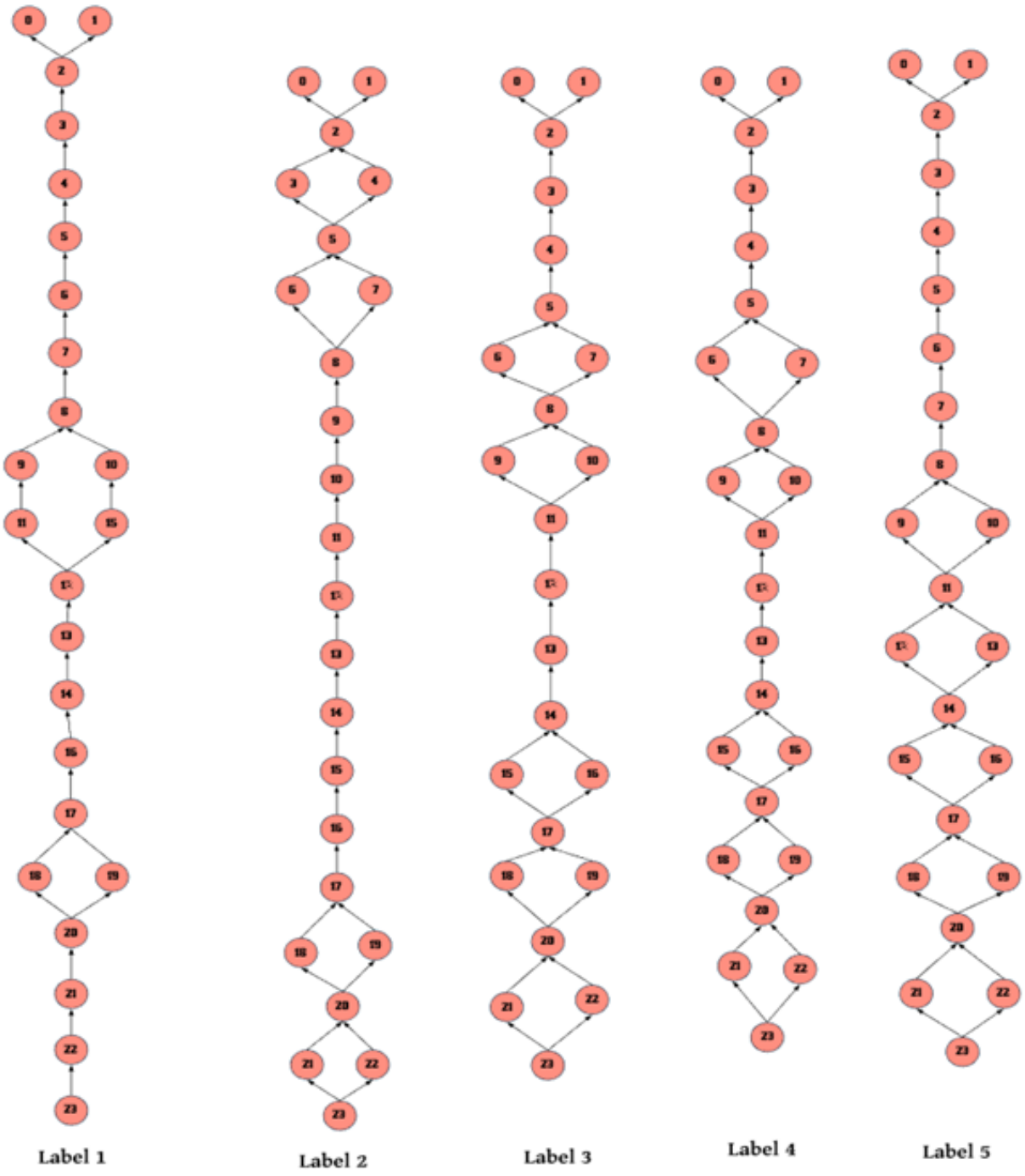

Figure 4.10: The DDT associated with the print sequence represented in Table 4.2.

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

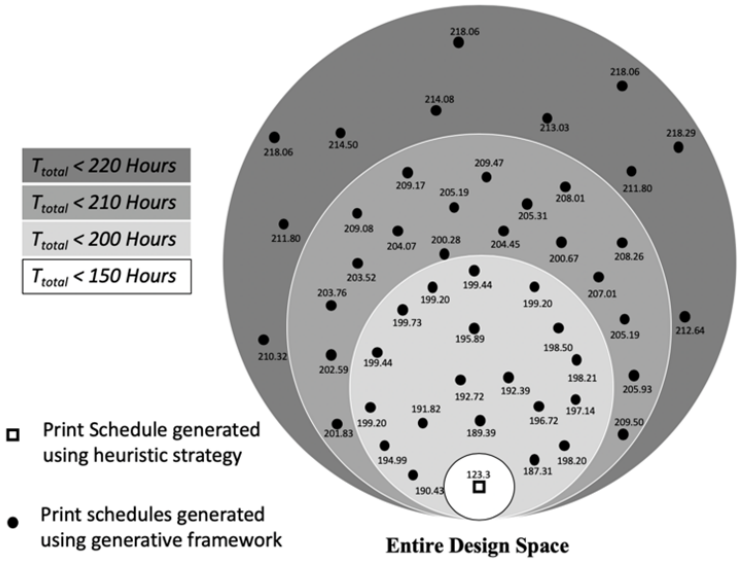

Figure 4.11: The plot showing the print time for each valid generated tree.

of optimization methods for the scheduling strategies.

Similarly, we plotted the total print time of every generated valid print schedule as well as the one generated from the heuristic approach for the second case study in a stacked Venn diagram, as presented in Figure 4.11.

4.5 Conclusion

In this chapter, 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 collisions 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 the 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 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 chapter are:

- 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 the chunker.

- Use of geometric constraints identified in our previous studies in chapter three to check the validity of the generated print schedules.

- Use of an evaluation time metric using a directed dependency tree (DDT) developed in the previous Chapter 3 to determine the total print time for each valid print schedule.

- Development of a generative approach that amalgamates print sequence generators, geometric constraint check, and time evaluation metrics that can automatically generate, validate, and evaluate a print schedule for a given chunking strategy.

- The approach presented in this chapter can be used to solve problems that couple production scheduling (IPPS) and pathfinding (MRS) with geometric partitioning (chunk-

ing) and changing geometric constraints 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 the next chapter, we will develop different approaches on top of the generative approach to find an optimal or near optimal collision-free print schedule that minimizes the printing time.

5 RESOURCE-CONSTRAINED SCHEDULING FOR MULTI-ROBOT COOPERATIVE 3D PRINTING

5.1 Overview

In Chapter 3, we developed a heuristic-based scheduling approach for C3DP. It is the first working schedule for C3DP. The heuristic approach was developed based on the assumption that the number of available robots is unlimited. In addition, the heuristic approach is limited by its optimality and generalizability. To overcome these limitations, we developed a generative approach to scheduling C3DP in Chapter 4. This approach can exhaustively search the design space to generate a variety of print schedules for a specified number of chunks with unlimited robots [4]. Such an exhaustive-search approach becomes intractable with the increase in the number of chunks. In this chapter, we present two C3DP scheduling methods that consider the resource constraints (e.g., with a limited number of robots) for C3DP scheduling, which can significantly reduce the search space for valid solutions. This work is a continuation of our prior study [68] and uniquely contributes to

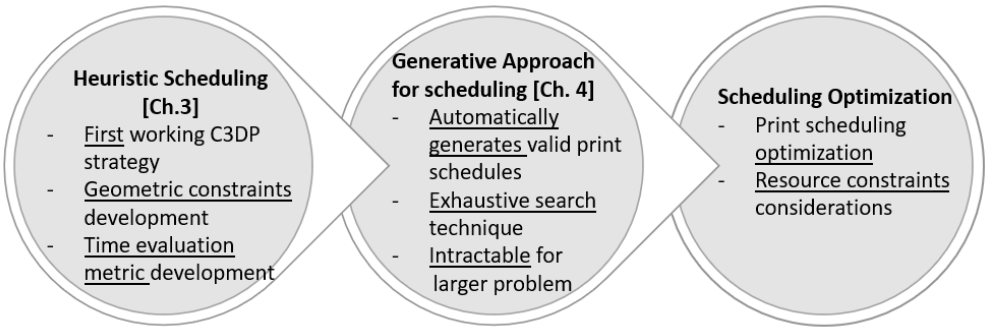

Figure 5.1: Summary of transition from our prior chapters to current one.

the C3DP literature in the following three aspects:

- Optimization: While the Chapter 4 focused on generating valid scheduling strategies, the current study in this chapter searches for the best scheduling strategy in the design space using the stochastic approach (i.e., the MGA-CC method) and a constraint-satisfaction approach (i.e., the DDLA-method).

- Tractability and Scalability: The generative framework presented in Chapter 4 exhaustively searches the entire design space, which is not tractable even for a small-scale problem. The study in this chapter reduces the size of the design space using two different methods: stochastic and constraint satisficing methods. They are capable of searching the design space of large-scale problems in a relatively short time.

- Resource Utilization: While our previous work in Chapter 4 allows users to specify the number of robots available for a job, the developed generative framework considers all possible schedules in the design space, even if some do not fully utilize all the available robots. The current study has a resource-constrained protocol to ensure that all the available robots will be fully utilized for printing.

The summary of our previous chapters, along with the scope of this chapter, is presented in Figure 5.1. In C3DP, the scheduling contains two dependent operations, 1) the chunk assignment, i.e., the allocation of chunks (subtasks) to the available mobile printers, and 2) the robot scheduling, which determines the order of the chunks to be printed (both in parallel as well as in series). The objective of this study is to develop scheduling methods for C3DP by taking constraints in the chunk assignment and print sequencing into consideration with limited resources. Such methods take geometric dependencies between chunks (gener-

ated from the chunking operation) as an input. It then assigns the chunks and schedules the robots to reduce the make-span.

The chapter is organized as follow:

5.2 Resource-Constrained Scheduling Problem Formulation presents the problem definition and provides mathematical formulation of the problem.

5.3 Methodology provides the detailed information of the two methods used to obtain multi-robots print schedule.

5.4 Performance Evaluation demonstrates the use of the two methods in two different case studies and also presents a detailed results of the demonstration and discussions associated with the results.

5.5 Conclusion outlines the concluding remarks and major lessons learned from the study.

5.2 Resource Constrained Scheduling Problem Formulation

In C3DP, the chunking operation partitions a part in n chunks represented as $C = c_1, c_2, c_3, ..., c_n$. A dependency list D describes the geometric dependencies between the chunks (e.g., one chunk sitting on top of another), where D is the output of the chunking and can be represented as $D = c_0 : [\phi], c_1 : [\phi], c_2 : c_1, c_3 : [c_0, c_1], ...$ The keys represent the chunks, and the values represent their dependencies. The dependency list provides information about which chunks can be printed first and which chunks must wait until their dependencies are completed. For example, c_0 and c_1 can be printed first because they have no dependency, whereas c_3 cannot be printed until both c_0 and c_1 are printed. For example, if we look at the three chunks in Figure 5.2. The dependency list can be represented as $0 : [\phi], 1 : [0], 2 : [0]$, meaning that chunk 0 must be printed first, followed

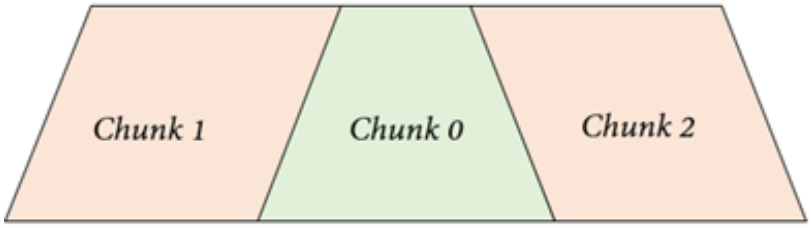

Figure 5.2: A part with 3 chunks. Chunk 0 must be printed prior to chunk 1 and 2.

by chunk 1 and chunk 2. But the dependency list itself is not enough to generate a print schedule, especially in the case where resources are limited. This is because it does not consider spatial constraints. The total number of available printing robots is m, represented as $R = r_1, r_2, r_3, ..., r_m$. The individual print time for each chunk is represented as $T = t_1, t_2, t_3, ..., t_n$. Each printing robot can only print one chunk at a time. Once the chunk is completed, the printing robot moves on to the next assigned chunk location. Since the print time of a chunk is much longer than the time it takes for printers to move from one location to another, the travel time is not considered for calculating the total print time. The objective function is, therefore, to minimize the print completion time of the chunk that is printed last (c_{ij}) and it is subjected to the following constraints:

1. Definition of c_{ij}

$$min\ c_{ij} = min(max(s_{ij} + t_j))$$

2. Dependency Constraints

$$s_{i,j} - s_{k,l} + t_l > 0, \forall\ j \neq l; i, k\ \epsilon\ R; \{c_j : [c_l]\}\ \epsilon\ D$$

3. R2R collision constraints (no collision between the active printing robots)

$$SV_i \cap SV_j = \phi \; \forall i \neq j; \; i, j \; \epsilon \; R$$

4. R2P collision constraints (no collision between the robot and the already printed part during travel)

$$R_i \cap P_j =, i \; \epsilon \; R, j \; \epsilon \; C$$

5. One chunk, one robot constraint (One chunk can only be printed by a single robot; a chunk must be completed before the printing robot can move to the next chunk for printing)

$$\sum_{i \epsilon R} \sum_{j \epsilon C} x_{ijk} = 1$$

$$\sum_{i \epsilon R} \sum_{k \epsilon C} x_{ijk} = 1$$

Variable's definition:

x_{ijk}: Binary variable,1 if robot i prints chunk j before chunk k, 0 otherwise

s_{ij}: Start time of chunk j on robot i

Notation definition:

$max(s_{ij} + t_j)$: Start time of last chunk j printed on robot i plus the print time of chunk j

$SV_{i,t}$: Swept volume of robot i at time t

$R_{i,t}$: 3D space occupied by robot i at time t

P_j: 3D space occupied by the printed chunk j at time t

$R : r_1, r_2, r_3 \ldots, r_m$, set of robots

$C : c_1, c_2, c_3 \ldots, c_n$, set of chunks

$T : t_1, t_2, t_3 \ldots, t_n$, print time of chunks

$D : c_0 : [\phi], c_1 : [\phi], c_2 : [c_1], c_3 : [c_0, c_1], \ldots,$ dependency list

5.3 Methodology

To address the scheduling problem in C3DP, new methods that take the following unique characteristics in C3DP are required.

- The R2R collision between printing robots while they are cooperatively printing assigned chunks.
- The potential R2P collisions between the printed part and the active robots.
- The dynamic environment due to the increase of printed volume over time.

In this section, we present two methods, dynamic dependency list algorithm (DDLA) and modified genetic algorithm with collision check (MGA-CC), to generate collision-free C3DP schedules, aiming to minimize the total print time under limited printing resources. Both methods significantly reduce the solution space by using a geometric dependency list as an input, which eliminates any solutions that violate the geometric dependency relationship between the chunks. In DDLA, the dependency list is used as a starting point to group non-dependent chunks (i.e., the chunks that can be printed together without collision) to generate a print sequence. It uses the FIFO (first in, first out) approach for the chunk assignment. By contrast, the MGA-CC randomly generates a population of initial chunk assignments and uses the dependency list in conjunction with the chunk assignments to generate print schedules. Genetic operators are then applied to modify the chunk assignments until a

specified criterion is met. Both DDLA and MGACC have a collision check incorporated, which checks for the potential R2P and R2R collisions.

5.3.1 Method 1: Dynamic Dependency List Algorithm (DDLA)

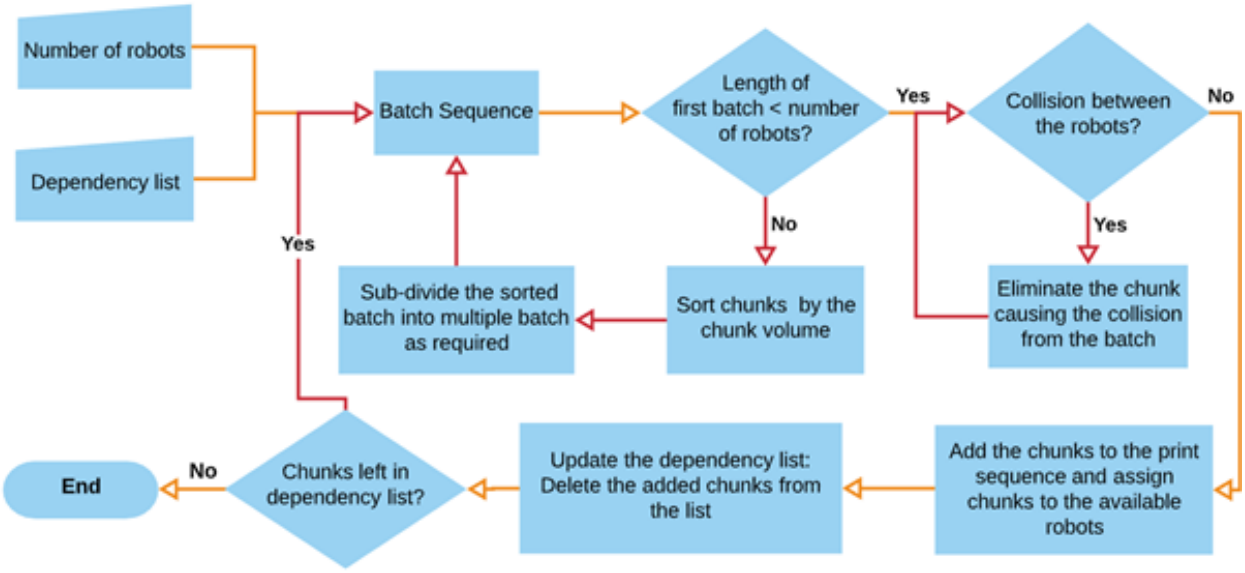

Figure 5.3: Flowchart showing the different step of DDLA.

The flowchart of the Dynamic Dependency List algorithm method is presented in Figure 5.3. Once a part is divided into smaller chunks, the chunks are indexed, and a geometric dependency list is created. In addition to the geometric dependency, the spatial constraints (geometrical dimension as well as the location of chunks) should also be considered to ensure that the print schedule is valid. For example, if referring to Figure 5.2, even if the geometric dependency is taken into consideration, a collision could still happen if chunks 1 and 2 are printed simultaneously. One way to make sure such a collision does not happen is to check the swept volume of robots printing the said chunks as presented in equation (5.1). Thus, before scheduling these two chunks in the same time slot, potential collisions between the

robots, while they are printing (not the collision while they are moving from one print location to another), need to be checked. To address this issue, the DDLA uses the following three steps:

1. Based on the dependency list, a batch sequence is created. A batch sequence is a list of chunks that can potentially be printed together. An example is shown in Figure 5.4, where the initial dependency list is presented in Figure 5.4(a). Based on this dependency list, a batch sequence is created, as shown in Figure 5.4(b). All the chunks that do not have any dependencies are grouped together to create a batch sequence. The total number of available robots is not considered while creating the batch sequence. Once the batch sequence is generated, we need to check for collision between the active robots to ensure that the chunks that are in the same batch can be printed together. To check for such collision, we use a swept volume of robots as outlined in Chapter 3.

$$SV_{R,i}(t) \cap SV_{R,i}(t) = \phi, i = 1, 2, 3 \ldots, n; j = 1, 2, 3 \ldots, n; j \neq i, \tag{5.1}$$

where $SV_i(t)$ is the swept volume of robot i, and $SV_j(t)$ is the swept volume of robot j at time t. If the intersection between the swept volumes of the printing robots is anything other than null, as presented in equation (5.1), there is potential for collision between the robots, and the chunks should not be printed at the same time.

2. Once the batch sequence is created, the algorithm checks whether the batched chunks can be printed without violating equation (5.1). Based on the number of printing robots (m) and the total number of chunks in the first batch (b), one of the following two scenarios will take place.

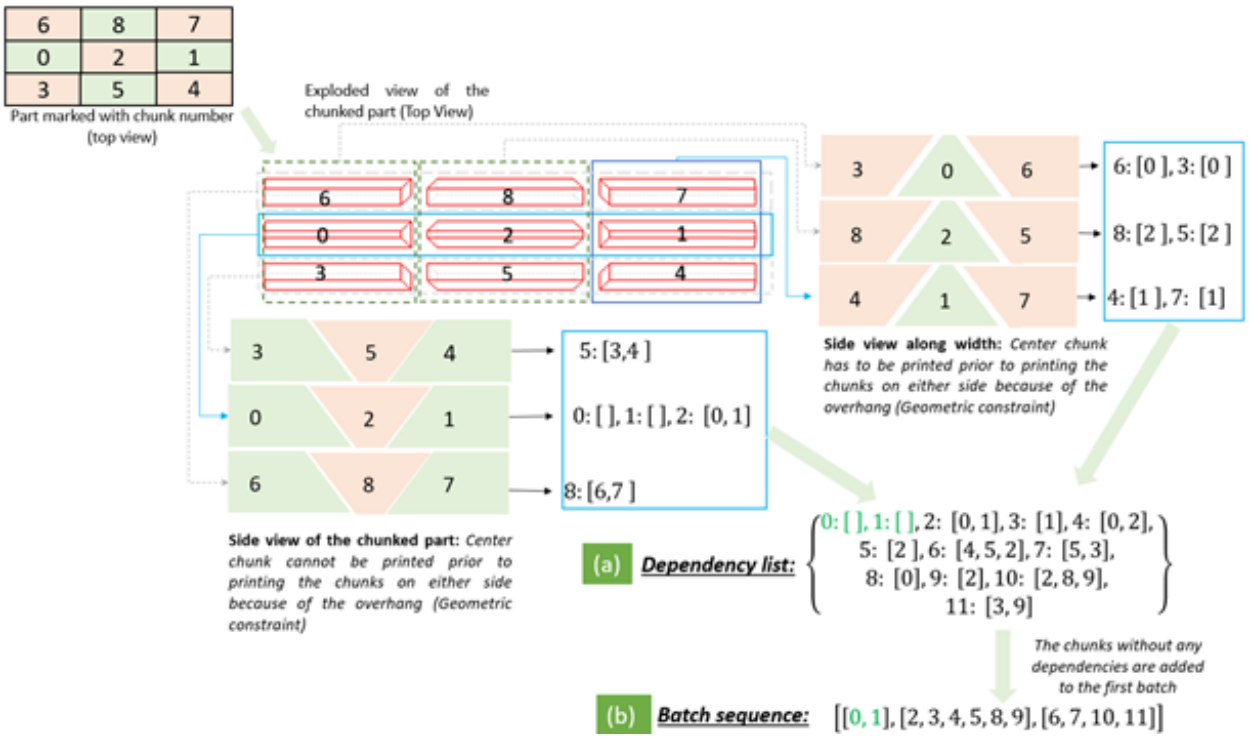

Figure 5.4: A part divided into 9 chunks (a) Resulting dependency list (b) Batch sequence from dependency list.

(a) If the number of available robots is greater than or equal to the number of chunks in a batch, i.e., $b \leq m$ and $SV_i(t) \cap SVj(t) = \phi$, the chunks are scheduled to be printed together. The chunk assignment takes place based on availability, i.e., the first robot available for printing is assigned to the chunk. For example, in the batch sequence in Figure 5.4(b), the first batch contains chunk 0 and chunk 1. Since no collision could happen when printing these chunks, they are added to the print schedule as a first print sequence.

(b) If the number of chunks in a batch is greater than the number of available robots, i.e., $b > m$, equation (5.1) will check which chunks can be printed simultaneously without collisions so that those chunks will be scheduled. A greedy approach is

then used to sort chunks based on print time in descending order. Doing so allows grouping larger chunks together. This ensures that each print sequence takes the shortest time to print by reducing idling or waiting. For example, if the first batch in a batch sequence has ten chunks, but only four robots are available for printing, first, a collision check is conducted. After that, if six of them can be printed together, the chunks are sorted based on print time in descending order. The first four chunks are chosen and added to the print schedule as the next print sequence. The remaining two chunks will be added to the rest of the chunks that are not scheduled yet. The two chunks will not be automatically scheduled next because, for each print sequence, we want to maximize parallel printing. On the other hand, if $b \ mod \ m = 0$ (i.e., the remainder of when the number of chunks in a batch is divided by the number of robots is equal to zero), and chunks can be simultaneously printed, then $b \div m$ sequences will be added to print schedules (i.e., the number of sequences added to the print schedule is equal to the result of the division of the number of chunks in a batch by the number of robots). For example, if eight chunks ($b = 8$) could be printed together without collisions, and four ($m = 4$) robots are available, two print sequences (each print four chunks at a time) will be added to the print schedule.

3. After a print sequence is added to the print schedule, the dependency list is updated by removing all the chunks that have already been added to the schedule. The dependency list is updated because doing so allows the algorithm to generate a print sequence with chunks that have their dependencies already satisfied.

This process is repeated until the full print schedule is created. The advantage of this method is that it is computationally efficient and scalable. Although the number of the print sequence is optimized using this method, the chunk assignment follows first in-first-out (FIFO) and may result in a suboptimal chunk assignment. This method is especially useful in the situation where each print sequence must be completed prior to starting the next print sequence, i.e., synchronous tasks.

5.3.2 Method 2: Modified Genetic Algorithm with Collision Check

The genetic algorithm has widely been used in both IPPS and MRS task allocation because it provides satisfactory performance for combinatorial problems [69]. Each solution in GA is represented in the form of a chromosome, so the first step is to encode the printing schedule in the form of a chromosome. Once encoding is done, the initial population will be generated, and then followed by evaluation, selection, and genetic operator application. The flow of the MGA-CC algorithm is presented in Figure 5.5.

1. *Encoding of individual chromosome* The chromosome representation for the C3DP task assignment is presented in Figure 5.6. If a part is divided into n chunks, there are n genes in the chromosome, and the index number of the gene is the chunk number. Each of the genes has a machine number encoded onto it, which corresponds to the robot number assigned to the chunk. For example, the chromosome in Figure 5.6 has zero encoded at index zero and two encoded at index one. This means that chunk zero is to be printed by robot zero, and chunk one is to be printed by robot two. Though the chromosome representing chunks assignment is necessary, it is not sufficient to determine the printing order of chunks because it does not include information

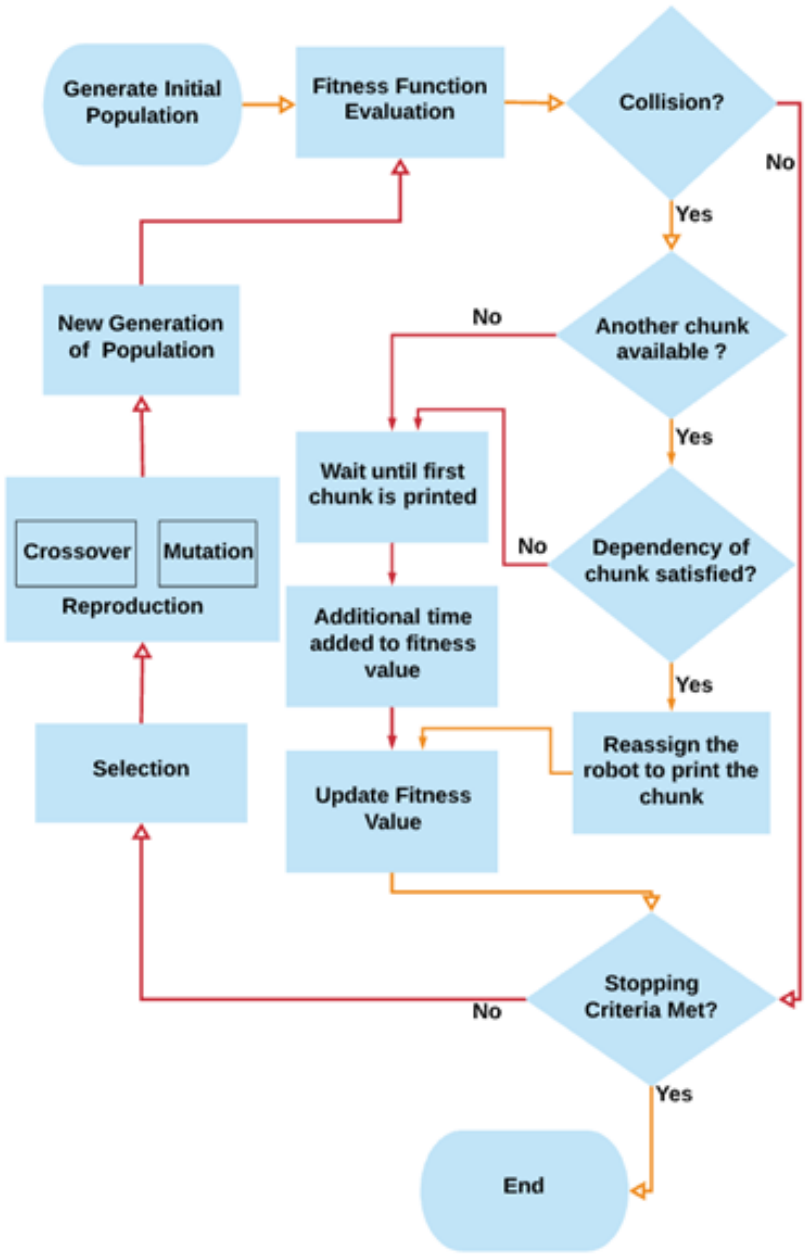

Figure 5.5: Flowchart showing the different step of MGA-CC.

regarding dependencies between the chunks. Thus, the chromosome is used with the dependency list to determine the printing order of the chunks.

2. *Random generation of the initial population*: The first generation of the population is generated randomly for a given number of chunks and the available robots. To generate

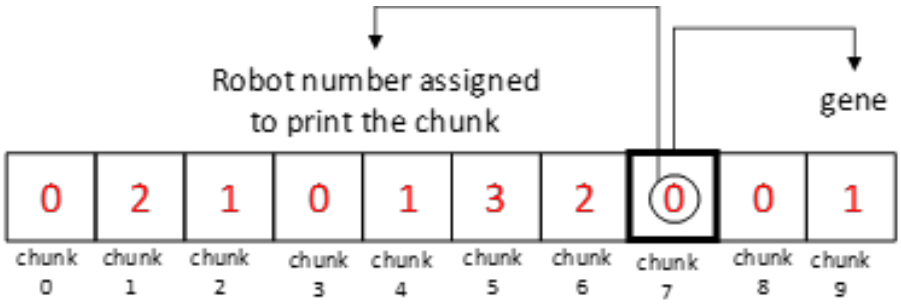

Figure 5.6: Chromosome encoding in C3DP.

a random chunk assignment, an empty array with the same size as the number of chunks will be filled with randomly generated robot numbers following a uniform distribution. This random gene generation approach is continued until the size of the population reaches the predetermined threshold for population size.

3. *Evaluation using a fitness function*: The fitness function is used to evaluate the performance of each chromosome. Since it is a minimization problem, the lower the value of the fitness, the better is the chromosome. The total path traveled between the print sequences is not taken into consideration. So, the fitness function solely focuses on the makespan through the assignment of chunks to the available printers. To determine the make-span, two things need to be considered: the dependency between the chunks and the next availability of the assigned robot. Since a chunk cannot start printing until all its dependencies are finished printing, Equation (5.2) can be used to determine when a chunk can be printed.

$$T_{start}(C_i) = max[T(C_1), T(C_2), T(C_3), \ldots\ldots, T(C_n)], \tag{5.2}$$

where,$T_{start}(C_i)$ is the start time for printing chunk i, $T(C_1)$ is the completion time for chunk 1, $C_1, ..., C_n$ are the dependencies of chunk C_i which are to be printed prior to starting the chunk.

A chunk might have all its dependencies satisfied but still might not be printed because the robot assigned to print that chunk might be in operation. Equation (5.3) takes both factors into account and calculates the start time of a chunk on the assigned printing robot, where M_j is the next available print time on the machine j, and $T_{start,ij}$ is the actual start time for the chunk i on the machine j.

$$T_{start,ij} = max(T_{start}(C_i), M_j) \tag{5.3}$$

The completion time of chunk i is given by equation (5.4), which is the sum of the start time of chunk i and the print time of that chunk.

$$T_{end,ij} = T_{start,ij} + T(C_i), \tag{5.4}$$

where, $T(C_i)$ the time it takes to print chunk C_i. Since the robots are assumed to be homogeneous, there is no difference in the completion time of the same chunk, even if a different robot prints it.

4. *Collision check*: Following equation (5.1), if there is a collision between the robots, one of the robots can be reassigned to print a different chunk. If no other chunks are available for printing, one of the robots must wait until the other robot finished printing its assigned chunk. The extra waiting time is added to the objective function

and works like a penalty value, which reduces the fitness value of the print schedule. So, its likelihood of being selected for the next round of population generation will be decreased. If there is no delay, the time is updated based on the reassignment and updated schedule, as shown in Figure 5.5.

5. *Genetic operation*: Three genetic operators are used in MGA-CC: selection, crossover, and mutation. First, chromosomes are to be selected for reproduction, and genetic operators of mutation and crossover are applied to them to generate the next generation of the population. Elitism is implemented to ensure that the fit individuals are passed on to the next generation of the population. Thus, the top twenty-five percent of the elite individuals are passed on to the next generation without the application of genetic operators to them. During every iteration, new chromosomes are added to the population to replace the chromosome with a low fitness value. This is done to avoid getting trapped in local optima.

6. *Selection method for individual*: There are different selection methods for reproduction to generate the next generation of the population. Some of the popular selection methods are tournament selection method [70], roulette method [71], rank-based selection method [72], and random selection method [72]. To keep the population diverse and avoid premature convergence, the random selection method is adopted at the beginning of the implementation, and the roulette wheel method is used towards the end of the implementation. In the random selection method, an individual chromosome is randomly chosen from the pool of the entire current population for reproduction. On the other hand, in the roulette wheel method, a fitter individual has a higher proba-

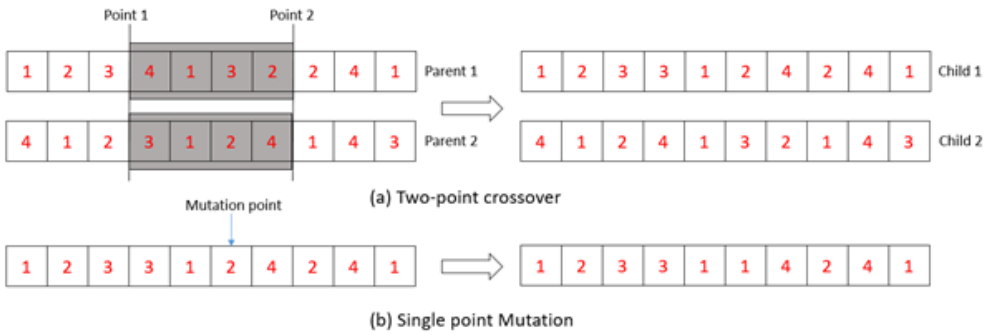

Figure 5.7: (a) Two-point crossover (b) Single point mutation.

bility of selection for reproduction. Once the parents are selected, genetic operations such as crossover and mutation are applied to them.

7. *Crossover operation*: Crossover requires two parents and produces two offspring. Two-point crossover is implemented in this study as the preferred choice of crossover. In the two-point crossover, two random points are selected in the chromosome, as shown in Figure 5.7(a). The segments between the selected points are swapped between the parents resulting in two children. Such crossovers reassign the chunks to different robots but do not change the order of the chunks.

8. *Mutation operation*: Mutation requires a single parent and result in a single offspring. The mutation operator is implemented as follows: Once a parent is selected, a mutation point is chosen randomly in the parent chromosome. The mutation point then undergoes a reassignment to a different robot for printing. This assignment is done randomly. The process is shown in Figure 5.7(b).

5.4 Performance Evaluation

In order to compare the two proposed methods, we demonstrate the application of the methods in three different cases in the C3DP setting. The cases range from a very simple part resulting in 20 chunks that are to be printed with four robots, as shown in Figure 5.9, to a much larger one, where a part results in 200 chunks that are to be printed using ten robots. In addition to this, a third case study presents a more complicated geometry with different chunk sizes and is to be printed using available robots, as shown in Figure 5.12. Two out of three case studies presented here (case study I and case study III) were also used to demonstrate the generative framework in Chapter 4. Although the two same case studies are adopted in the current work, it is fundamentally different from our prior because this is the first time, we have developed methods that seek to minimize the makespan for C3DP scheduling with limited resources. To simplify the problem, we make the following assumptions for all case studies that are presented in this section.

1. The chunking process is not part of the scheduling process and is predetermined. The desired part is provided to the chunker [3], which outputs required information for task assignment and scheduling such as chunks dependencies, coordinates of the chunks, etc.

2. The number of available robots is defined by a user.

3. The path planning is not considered, and thus, any collision that might occur while the robots are traveling from one print location to another is not considered as well. The robots spend most of the time printing (roughly ¿ 95%) compared to traveling. Moreover, the collision-free path planning for multiple robots is an NP-hard problem.

It may change the entire problem to a multi-level decision-making problem, which is worthy of a separate study in and could be part of the future work.

To evaluate the proposed approaches, the quality of the solution (how good the total print time is compared to one another) and the execution time were used as evaluation metrics. Since the MGA-CC is a stochastic approach, measuring the actual runtime might give a better idea of the scalability of the approach. Therefore, the comparison between the two algorithms is presented in terms of execution time rather than the number of function evaluations or the number of iterations.

5.4.1 Benchmark Test

Conventionally, the scheduling of multiple jobs in multiple machines is formulated as a mathematical problem. The problems are then solved using exact methods, which include commercial solvers such as cplex, or some meta-heuristic techniques that might not provide exact results but are computationally less taxing than the exact methods. The exact methods include dynamic programming methods, which try to find the optimal schedule by complete enumeration, or linear programming-based approaches such as MILP (mixed-integer linear programming) that used the branch-and-bound algorithm. While both of these approaches provide the exact solution, they are computationally expensive and do not scale to larger problems. For example, the approach can easily solve the problem presented in case studies I and III; however, case study II is too large to solve within a reasonable timeframe. But we can use them as benchmark tests to solve the multi-robot scheduling problem in C3DP for case studies I and III to compare the results of the two approaches presented in the chapter with that of exact methods.

Thus, we use the MILP solver, which implements an LP-based branch-and-bound algorithm to solve the problem and compare the results with the ones from the proposed approaches. However, aside from the issue of scalability, another issue plagues the implementation of MILP for C3DP scheduling – the MILP lacks the ability to incorporate dynamic constraints, where the chunks are printed by the assigned robots in a sequence that is not known beforehand. This makes it difficult to add dynamic collision avoidance constraints. To circumvent this issue, the chunks that would cause robots to collide in printing were identified first manually, prior to the implementation of MILP. Once identified, constraints were added individually to avoid schedules that result in such collisions. However, case study II is too large to manually add constraints because it contains 200 chunks and ten robots. Thus, no benchmark test is presented for case study II.

5.4.2 MGA-CC Parameters

The performance of evolutionary algorithms such as MGA-CC largely depends on the configuration values for mutation and crossover operations. In addition, the performance also depends on how many chromosomes are generated for the population. Thus, it is important that the parameters are carefully chosen, but since the problem of C3DP is novel, there are no standard values that could be adopted from the literature. Thus, to determine the proper values of the different parameters, a sensitivity analysis was conducted. For the crossover rate, an initial value of 0.10 was chosen and increased by 0.1 for every new data point. Similarly, an initial mutation rate of 0.05 was chosen and was increased by 0.01 for every new data point. Once the initial values and the increment were determined, multiple runs of the algorithm were carried out with different combinations of the sampled values.

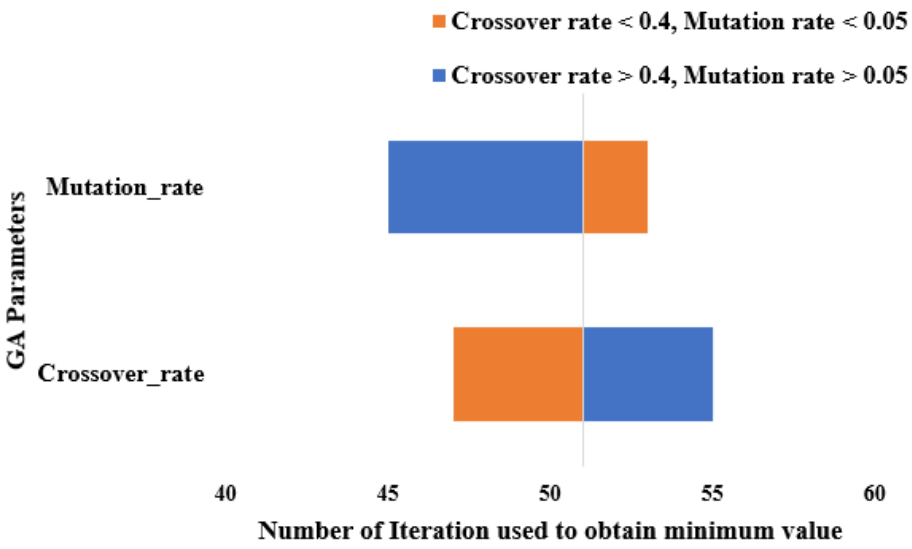

Figure 5.8: Simple sensitivity analysis for MGA-cc parameters.

The number of iterations required to obtain a minimum solution was used as the evaluation metric. Figure 5.8 shows the change in metric (number of iterations required to obtain the solution) in response to the independent parameters' baseline value (i.e., the crossover and mutation rates). A crossover rate of 0.40 and a mutation rate of 0.05 were used as initial configurations to create the graph. It can be observed that if the mutation rate is halved to 0.25, it has a larger significance (results in a larger change in the used metric) compared to if the mutation rate was doubled to 0.10. On the other hand, the change in the number of iterations required to obtain a minimum solution is not as sensitive to the crossover rate change. For the population size, a general rule of thumb is that a smaller population size provides quicker convergence, albeit a caveat that it might get trapped in the local minima. Since the case studies involve two smaller problems and one medium-scale problem, the population size of 50 was chosen, which provides a good balance between diversity and the computational efficiency of the algorithm. Based on the result of the tests, the population

size of 50, the mutation rate of 0.05, and the crossover rate of 0.40 are adopted for all three case studies. The algorithm typically converges after 500 iterations for every case study.

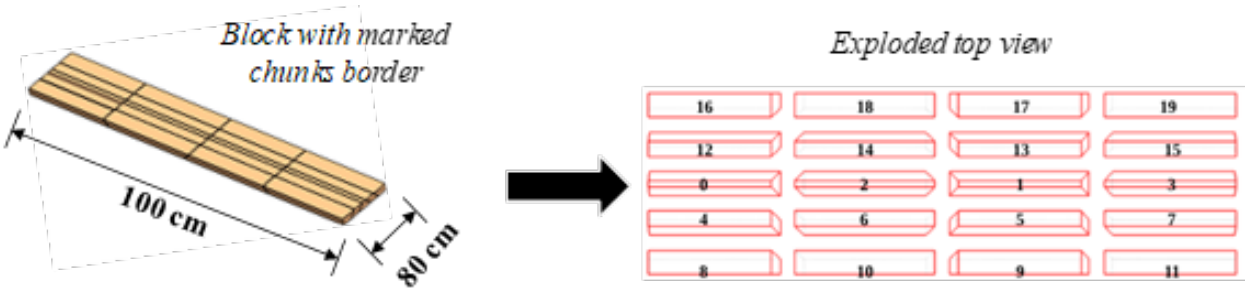

Figure 5.9: Rectangular block and exploded view of the chunks.

5.4.3 Case Study I: 20 Chunks and Four Printing Robots

For the first case study, a simple rectangular bar is divided into twenty chunks, with four columns and five rows. Four robots are available for printing in this case. The rectangular bar, with its dimension along with the chunks, is presented in Figure 5.9. Since the path planning is not considered, there are possible multiple optimal chunk assignments. In this case, the chunk volumes are homogenous, i.e., each chunk takes the same amount of time to be printed. Using a print speed of $16mm^3/s$, it takes the printer roughly about 10.42 $hours$ to print a chunk. The resulting chunk assignment, as well as the print schedules, are presented in Table 5.1 along with other pertinent information such as the total make-span and the time it took the methods to achieve the solution. Although both methods generate the same print schedule, the chunk assignment is different. The chunk assignment and the print schedule obtained using DDLA are presented in Figure 5.10(a), and the one obtained using MGA-CC in Figure 5.10(b). The print schedules are represented using a directed dependency tree (DDT) where the node represents a chunk to be

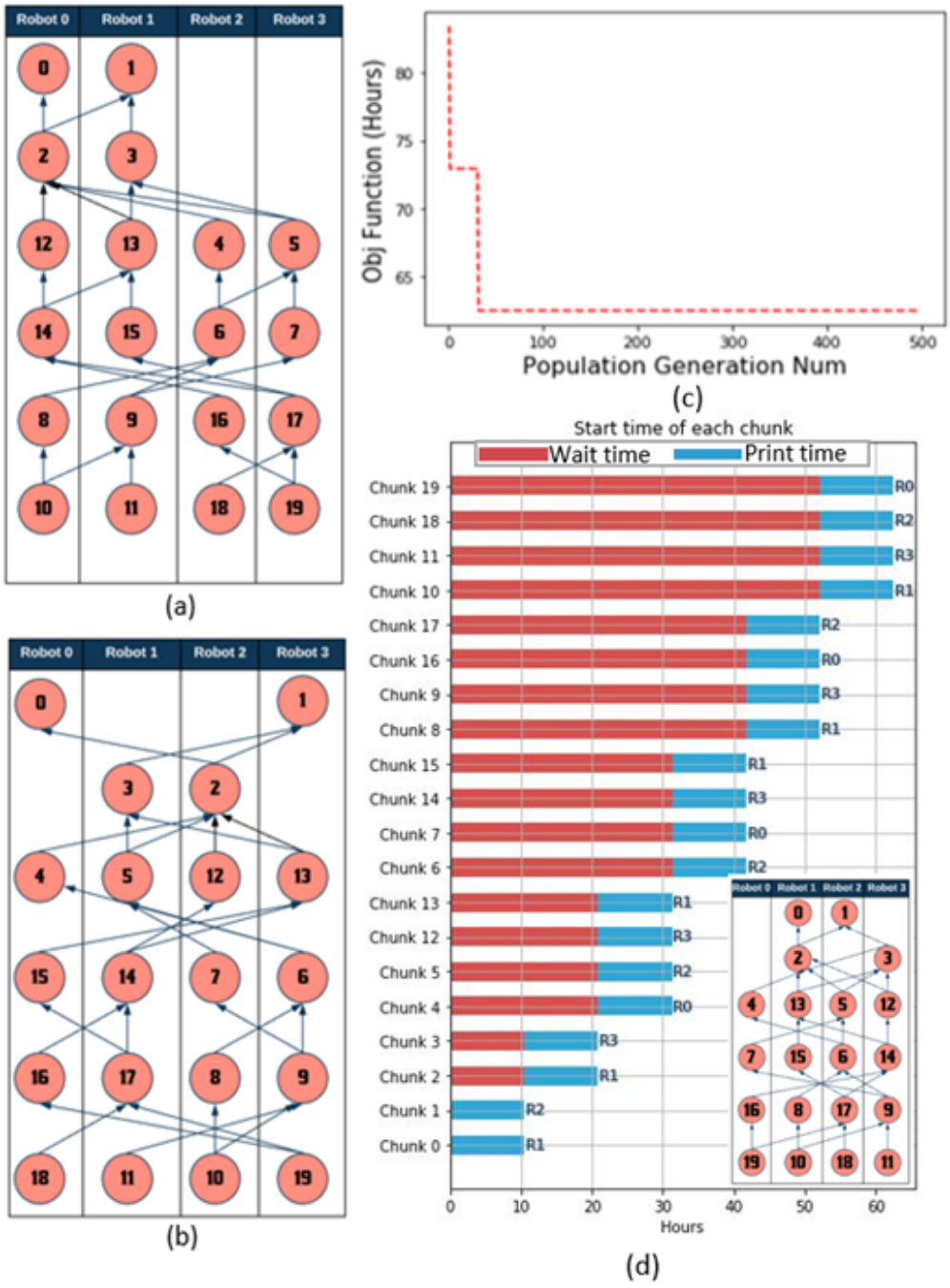

Figure 5.10: (a) DDT of print schedule using DDLA (b) DDT of print schedule using MGA (c) The change of objective function value with the new iteration of population generation in modified GA (d) The print schedule using MILP.

printed, and the edge represents a dependency relationship between the two connecting chunks. It can be observed in both print schedules that only two robots are used for printing

Table 5.1: Comparison between DDLA and MGA Case study I)

	DDLA	**MGA-CC**
Chunk Assignment	Robot 0: [0, 2, 12, 14, 8, 10], Robot 1: [1, 3, 13, 15, 9, 11], Robot 2: [4, 6, 16, 18], Robot 3: [5, 7, 17, 19]	Robot 0: [0, 4, 15, 16, 18], Robot 1: [3, 5, 11, 14, 17], Robot 2: [2, 7, 8, 10, 12], Robot 3: [1, 6, 9, 13, 19]
Print Schedule	{1: [0, 1], 2: [2, 3], 3: [12, 13, 4, 5], 4: [14, 15, 6, 7], 5:[16, 17, 8, 9], 6:[18, 19, 10, 11]}	{1: [0, 1], 2: [2, 3], 3: [12,13, 4, 5], 4: [14, 15, 6, 7], 5:[16, 17, 8, 9], 6:[18, 19, 10, 11]}
Make-span	62.52 Hours	62.52 Hours
Computation Time	<0.10 seconds	0.132 seconds

the chunks, whereas all four robots are used after the first two print sequences. The dependency list $(0:[\phi], 1:[\phi], 2:[0,1], 3:[1], 4:[0,2], 5:[1,2,3], 6:[4,5,2], 7:[5,3], 8:[4,6], 9:[5,6,7], 10:[8,9,6], 11:[9,7], 12:[0,2], 13:[1,2,3], 14:[2,12,13], 15:[3,13], 16:[12,14], 17:[13,14,15], 18:[16,17,14], 19:[17,15])$ that was used as input allows only two (chunk 0 and chunk 1) chunks to be printed first because only chunk 0 and chunk 1 have no dependencies that need to be satisfied. Once they are printed, the chunk that has chunk 0 and chunk one as dependencies can be printed next (chunk two and chunk 3). Both methods result in a make-span of $62.52\,hours$. While the DDLA method took $0.1\,seconds$ in the print schedule and assignment, MGA-CC took about $0.132\,seconds$. This is due to the fact that MGA-CC has to run 500 iterations of population generation even though it converged in less than 100 iterations, as presented in Figure 5.10(c). Even though the DDLA uses heuristics to assign the chunks to the robots, the homogeneity of the chunk printing time allows all the robots to complete printing their assigned chunk simultaneously. Thus, no robots must wait to start the next print sequence. And that is the reason the make-spans for both methods are the same. If the chunks are non-homogeneous, the make-span will likely differ, as observed in case study III. Finally, MILP calculated the optimal make-span of $62.52\,hours$ as well.

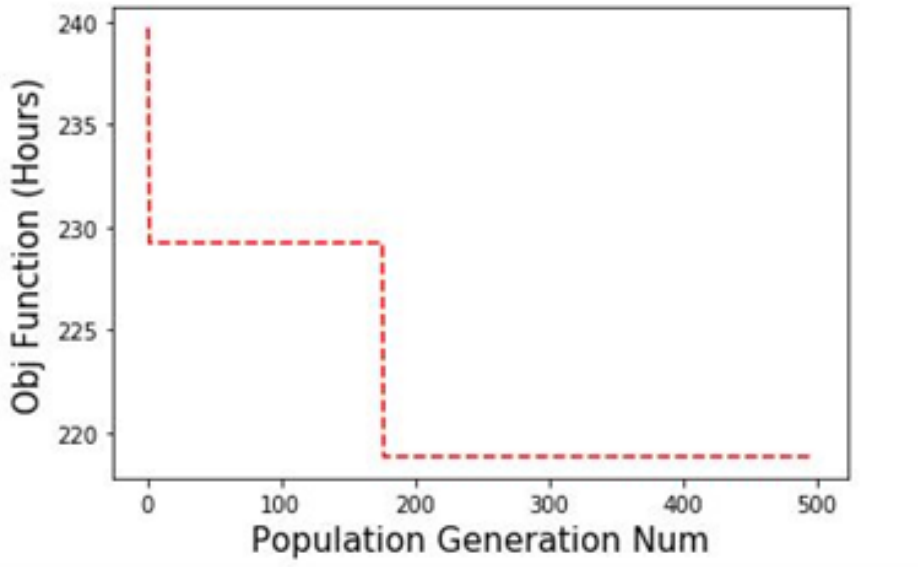

Figure 5.11: The change of objective function value with the new population generation in modified GA.

Though the schedule generated by MILP is the same as the one generated by DDLA and MGA-CC, as shown in Table 5.1, the chunk assignment is different. The schedule and the assignment obtained using MILP are presented in Figure 5.10(d). MILP took 6.97 *seconds* to get the said solution, making it slower than both the DDLA and MGA-CC.

5.4.4 Case Study II: 200 Chunks and Ten Printing Robots

For the second case study, a part is chosen that has a similar geometry to the first case study, but the size of the part is much larger ($500cm \times 80cm \times 1.5cm$), thus resulting in 200 chunks (5 rows and 40 columns). Each chunk takes about the same time to print as in the first case study ($10.42 hours$ printed at $16mm^3/s$ print speed). For this printing task, ten printing robots are available for printing, though ideally, 40 robots would be required for printing the part. Thus, the resources available are one-quarter of what is ideally required for the print

Table 5.2: Comparison between DDLA and MGA-CC Case study II)

	DDLA	**MGA-CC**
Make-span	218.82 hours	218.82 hours
Computation time	0.208 seconds	1.432 seconds

job. Since this is a very large print job with 200 chunks, outlining the entire schedule and chunk assignment for 200 chunks takes a larger space. Thus, the print schedule and chunks assignment are not presented, and only the make-span and computation times are provided in Table 5.2. Both DDLA and MGA-CC calculated the make-span to be 218.82 *hours*, but as observed in the smaller scale problem in case study I, the chunk assignment is quite different. Again, this can be attributed to the homogeneity of the chunks where chunks can be printed by the assigned robots in a synchronous manner, where all ten robots can finish printing the previous chunks and start printing new chunks together simultaneously. While DDLA took 0.208 *seconds* to compute the make-span, the MGA-CC took more than $6X$ the time to compute the exact same make-span. But as observed in Figure 5.11, the MGA-CC converged to the solution in less than 200 iterations.

5.4.5 Case Study III: 24 Chunks And Four Printing Robots

For the third case study, a part with more complicated geometry is chosen to test the generality of the methods. We use a toy folding SUV with a dimension of $157.6cm \times 140cm \times 3.6cm$. Since the part is larger compared to the first case study, we use a larger print speed ($40mm^3/s$ using a print nozzle of $1mm$ diameter) to reduce the print time for the individual chunk. The part and the resulting 24 chunks are presented in Figure 5.12. The part is divided into three columns and eight rows, but it can be observed in Figure 5.12(c) that the chunks in each row are very different in terms of size and shape and, thus, will have different

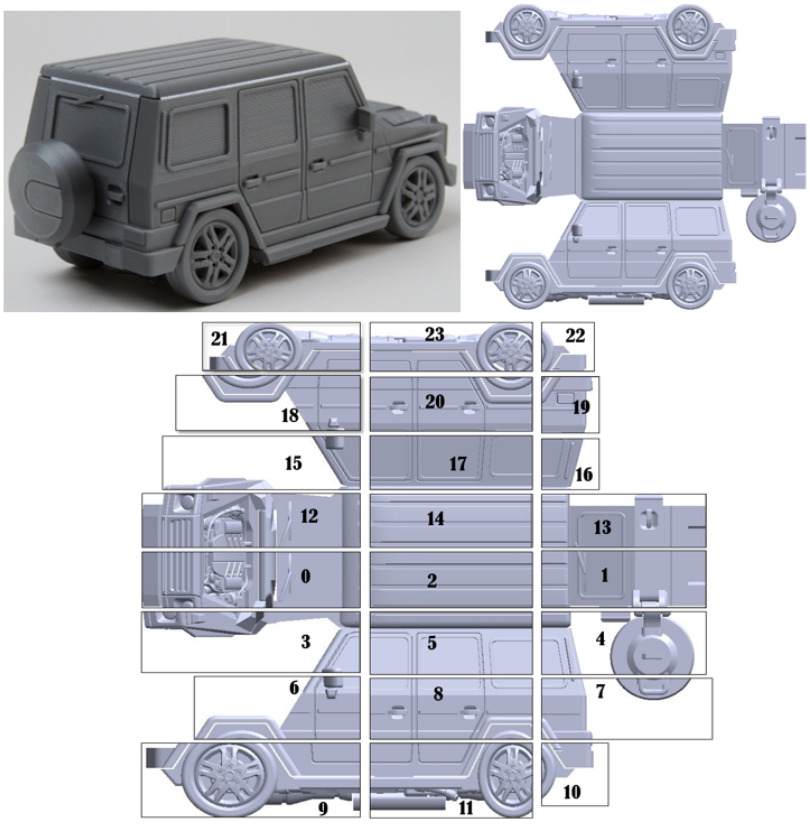

Figure 5.12: Folded 3D model of a printed SUV vehicle (Top Left). Top view of unfolded STL model of a SUV vehicle (Top Right). Rectangular block showing the chunks line, exploded view of the chunks that combine to make a rectangular block and top view of exploded chunks(Bottom).

printing times. Four robots are available for printing the part. Table 5.3 presents the chunk assignment and print schedule generated using both DDLA and MGA-CC along with the make-span and time it took each method to compute the solution. In addition, the chunk assignment, along with the print schedule represented using DDT, is presented in Figure 5.13. The DDT shows that even though the print sequences or the order in which chunks are printed remains the same for both methods, the assignment is quite different from one another, and the makespan is different for that reason. This is expected as DDLA generates print sequence first based on the dependency list and assigns chunks to robots based on FIFO.

Table 5.3: Comparison between DDLA and MGA-CC(Case study II)

	DDLA	**MGA-CC**
Chunk Assignment	Robot 0: [0, 2, 13, 5, 8, 16, 19,11, 21, 23], Robot 1: [1, 3, 14, 15, 17, 9, 20, 22], Robot 2: [4, 6, 10], Robot 3: [12, 7, 18]	Robot0: [0, 2, 3, 6, 16, 18, 23], Robot1: [6, 10, 11, 13, 14, 17], Robot2: [1,7, 12,19, 20, 21], Robot3: [4, 5, 8, 9, 15, 22]
Print Schedule	{1: [0, 1], 2: [2], 3: [13, 3, 4, 12], 4: [5, 14], 5: [16, 15, 6, 7], 6: [8, 17], 7: [19, 9, 10, 18], 8: [11, 20], 9: [21, 22], 10: [23]}	{1: [0, 1], 2: [2], 3: [13, 3, 4, 12], 4: [5, 14], 5: [16, 15, 6, 7], 6: [8, 17], 7: [19, 9, 10, 18], 8: [11, 20], 9: [21, 22], 10: [23]}
Make-span	114.17 hours	106.04 hours
Computation Time	0.15 seconds	0.326 seconds

It lacks a sophisticated approach required to track overall available times on all robots and assign chunks based on such availability. And though the homogeneity of chunks allowed the method to overcome such necessity in the first two case studies, the heterogeneity of chunks in this case study exposes its weakness. Though a more sophisticated approach of chunks assignment could be added to the method, it will impact the computational efficiency of the method, making it computationally slower. On the other hand, this is where MGA-CC demonstrates its superiority as it can find a near-optimal chunk assignment based on a given dependency list compared to the DDLA method. The make-span for MGA-CC was much shorter (106.04 $hours$) compared to that of DDLA (114.17 $hours$). The graph presented in Figure 5.13(c) shows the objective function value change over the new population iteration for MGA-CC. Thus, the MGA-CC was able to obtain a shorter make-span in slightly longer computational time and shows that it is better fitted for the C3DP scheduling problem where chunks have different shapes and sizes, resulting in different print times. Since the

scale of the problem is small enough, where manually adding dynamic constraints to avoid a collision is manageable, MILP was used to solve the problem as well. The optimal make-span obtained using MILP was 106.04 $hours$, which is the same as the one obtained using MGA-CC. While it only took MGA-CC 0.326 $seconds$, it took MILP over 10 $seconds$ to obtain the same solution. The schedule and the assignment obtained using MILP are presented in Figure 5.13(d). As mentioned, case II is a scaled-up version of case I, where case I have 20 chunks, whereas case II has 200 chunks. And between the two case studies, the computation time for MGA-CC increased by $10X$. On the other hand, DDLA is a deterministic approach, and between Cases I and II, the computation time goes up by $2X$. Thus, the complexity of DDLA is bounded by $O(dnm)$, where d is the number of discrete bins in the dependency list, n is the number of tasks or chunks, and m is the number of robots.

5.4.6 Comparison with The Heuristic Strategy

In Chapter 3, we proposed a heuristic-based print strategy and demonstrated it using a rectangular bar in chapter four. The same geometry is used as the first case study. The comparison of the heuristic print strategy (Figure 4.9(a)) with the one obtained using the two methods shows that they are the same print schedule. However, the chunk assignment that is based on the number of available printing resources plays an important role in determining the actual total print time. For example, in case study II, where the part has a larger number of chunks, but ten robots are available, while 40 robots would be ideal for printing, the heuristic print strategy could still work well; but since the printing resources are limited, only a certain number of chunks can be printed in parallel. Thus, in such cases, the chunk assignment plays an important role in reducing the total print time.

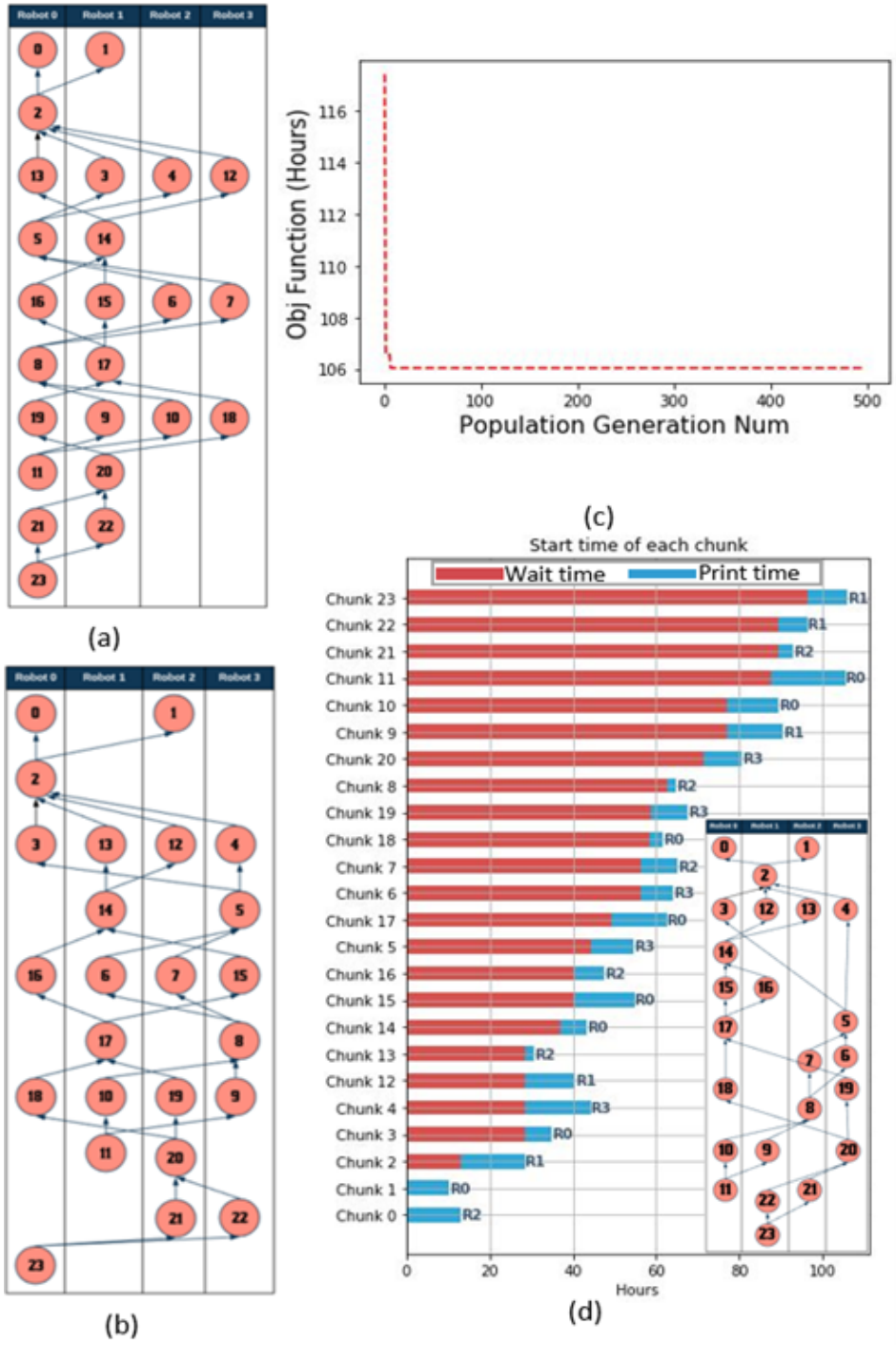

Figure 5.13: (a) DDT of print schedule along with the chunk assignment obtained using DDLA (b) DDT of print schedule along with the chunk assignment obtained using modified GA (c) The change of objective function value with the new iteration of population generation in modified GA (d) The print schedule along with chunk assignment obtained using MILP.

Similarly, the folding SUV used in the third case study was also used in study presented in Chapter 4, for which the print schedule was generated using human heuristics (Figure 4.9(b)). The print schedule generated by both methods is the same as the one generated using heuristics, but the total print time is longer for the heuristic print strategy (123.30 *hours* for heuristic vs. 106.04 *hours* for MGA-CC). This is because the heuristic approach does not consider the chunk assignment. A proper chunk assignment minimizes the waiting time and reduces make-span.

5.5 Conclusion

This chapter presents two methods for resource-constrained multi-robot cooperative 3D printing scheduling, i.e., the Dynamic dependency list algorithm (DDLA) and the modified GA with collision check (MGA-CC). Both methods use the dependency relationship between chunks as input but are different in searching design space. To demonstrate and compare the performance of these two methods, three case studies are conducted. For the first two case studies with homogeneous chunks in terms of printing time, both methods generated a print schedule with the same make-span. For the third case study, however, the MGA-CC method resulted in shorter overall print time even though the print schedule was the same for both methods. This can be attributed to the chunk assignment, where MGA-CC reduces the wait time of the robots by tracking their availability and thus, reducing the make-span. The DDLA method is expected to work best for synchronous tasks, where multiple chunks can be started together and completed together (e.g., shape formation). That is why DDLA was able to match the MGA-CC for the first two cases and not do the same for the third case study.

The proposed approaches for multi-robot scheduling lack such path planning and motion planning and do not yet demonstrate their interaction with scheduling. This is the limitation of the study presented in this chapter. Thus, to overcome this limitation and to take the next step in creating an autonomous manufacturing factory, a study of a scalable algorithm for scheduling along with path planning for the multi-robot manufacturing system is needed and is part of the next chapter.

6 CENTRALIZED VS. DECENTRALIZED PLANNING FOR COOPERATIVE 3D PRINTING

6.1 Overview

In the previous chapter, we presented a resource-constrained scheduling strategy for C3DP, which included a demonstration of two approaches to generate chunk assignments and print sequences in a cooperative 3D printing context, where printing resources were limited. While the presented approaches provided near-optimal results for problems with different scales, they did not account for path planning for the generated print schedule. Generation of a path for a given schedule is a non-trivial task for a multi-robot system but should be calculated, along with scheduling, to get a better approximation of the time it will take robots to finish printing a given job. The word approximation is used intentionally to highlight those inherent uncertainties that exist in manufacturing, especially additive manufacturing, and that a planned schedule and path rarely match the actual implementation.

Thus, to address this gap, in this chapter, we focus on developing multi-robot planning, which includes schedule generation as well as path planning for the robots. In addition to this, we also demonstrate a decentralized approach to C3DP planning, where each robot act as an autonomous agent. These agents make a decision based on the local information available to them. Contrary to the centralized approach, decentralized planning does not require planning for printing (schedule and path generation) prior to implementation. The decentralized approach is based on simple sets of rules and is inspired by nature, where ants follow the set of rules to work together. We demonstrate the newly developed approach

using two case studies and then compare the results with that of the centralized approach. These results are compared based on criteria such as makespan and runtime.

The chapter is organized as follows:

6.2 Introduction and Background provides background information regarding the centralized and decentralized approach in the context of C3DP.

6.3 Motivating Example demonstrates an example where two hexagons are to be printed using two robots and highlights uncertainties present in additive manufacturing and highlights a need of an approach that can handle such uncertainties.

6.4 Relevant Research discusses the application of the two approaches in the broader multi-robot research area

6.5 Approaches to C3DP Manufacturing provides more detailed information about each of the approaches

6.6 Results and Discussion presents the experimental results of the implementation using two case studies, including a rectangular bar with a variable number of chunks and a razorback mask.

6.7 Conclusion outlines the concluding remarks and major lessons learned from the study.

6.2 Introduction and Background

Cooperative 3D printing (C3DP), as illustrated in Figure 6.1, is a new form of the additive manufacturing process that uses multiple mobile 3D printing robots to accomplish large-scale printing tasks. Chunked-based printing [1] is one of the manifestations of multi-robot cooperative 3D printing, where a part is first divided into multiple chunks, and the chunks are then assigned to multiple robots to print simultaneously, thus reducing printing

time. Cooperative 3D printing overcomes the limitations of conventional 3D printing, such as slow printing speed and the inability of printers to print a part larger than themselves. The efficiency of cooperative 3D printing requires careful coordination among the robots to complete the printing tasks. Such coordination requires the available printing robots to work in parallel when possible, and avoidance of collision with other printing robots and printed materials, that can happen in an environment that is changing both in space and time. There are generally three approaches to solving the multi-robot coordination problem: centralized, decentralized, and hybrid approaches. Centralized approaches require a central planner responsible for planning the actions of all robots and communicating with individual robots. A more detailed review of recent literature using the centralized approaches is presented in Chapter 2. On the other hand, decentralized approaches involve no central planner, and the planning responsibility is distributed among all the independently operating robots that rely solely on information accessible to individual robots. The prominent differences between the two approaches are highlighted in Chapter 2 in Table 2.2. While both centralized approaches and decentralized approaches have been widely studied in the multi-robot systems literature over the past decades, the differentiation between the two approaches in the context of multi-robot manufacturing is not quite pronounced. Moreover, the use of decentralized approaches to cooperative additive manufacturing has not yet been investigated.

In Chapter 5, we implemented a centralized approach (Modified Genetic Algorithm and mixed-integer linear programming) to solve the multi-robot coordination problem for C3DP because centralized approaches allow easier integration of complicated problems structures such as the inclusion of manufacturing constraints and spatiotemporally changing environment. The use of such a centralized approach provided better control of the system as

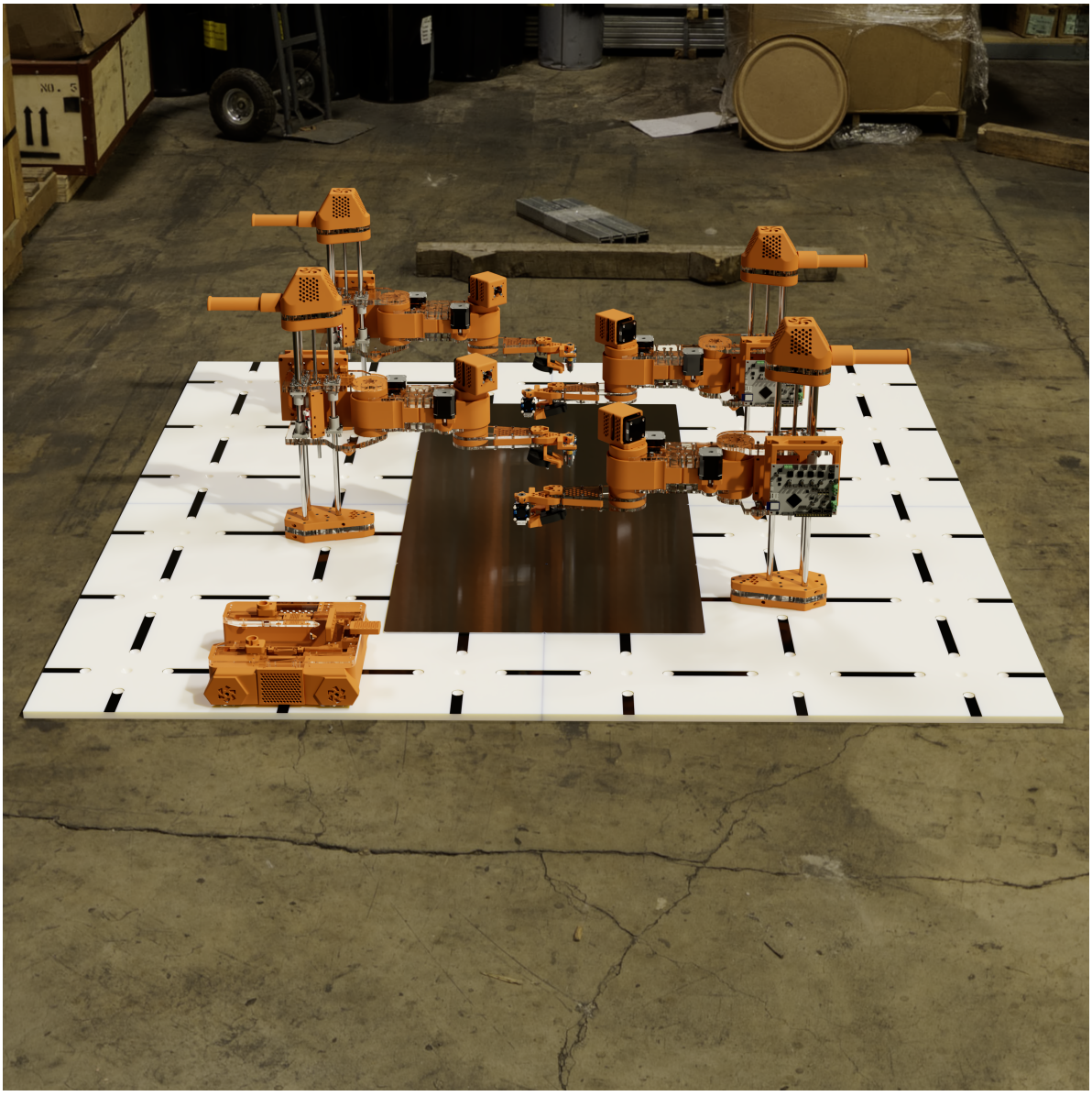

Figure 6.1: Demonstration of multi-robot Cooperative 3D printing where multiple robots are simultaneously printing in cooperative manner.

the status of the entire team of robots is known all the time. Using the approach, we were able to obtain near-optimal results for both small-scale problems and large-scale problems. But as C3DP is gradually adopted by the wider manufacturing community and print jobs become larger and more complicated, we might see a manufacturing floor extending in large distances, where many mobile robots move around the factory floor to accomplish multiple print jobs. The planning of such a system becomes complicated as the complexity of printing dependencies will grow with the number of robots and printing tasks.

While the application of the centralized approach has been proven to give very good results for a small-scale version of such a problem, could it be a bottleneck in large-scale C3DP? The centralized approach requires a robust communication scheme between the central planner and the entire team. Can that still be established on such a large scale reliably with reasonably low cost? In addition, as the number of robots increases, uncertainties in executions increase because it is difficult to pre-determine the timing of the execution of commands. There is also a high likelihood that robots might often fail as the overall environment become more dynamic. Such problems make it difficult to plan for multi-robot coordination using a centralized approach. In such cases, a decentralized approach could provide a better solution. Though a decentralized approach cannot provide a theoretical guarantee for optimal global solutions and may often be far from optimal in both path planning and resource allocation, does it work, without needing an expensive, robust communication framework for the large system? These questions are not clearly answered in manufacturing using multiple mobile robots. We aim to investigate the application of the two paradigms in the context of both large and small-scale manufacturing and understand the pros and cons, such as feasibility and scalability, of each paradigm in C3DP. This motivates us to explore and develop a

new decentralized multi-robot planning method in cooperative additive manufacturing and compare its performance with that of a centralized approach.

In this chapter, we introduce a decentralized approach for C3DP that takes inspiration from nature. The decentralized approach, swarm printing, is a framework where simple rules are formulated, similar to the traffic rules that humans follow to maintain structure while driving. Each agent (robot) adheres to these rules and coordinates with each other based on the local exchange of information. These agents are unaware of the global information, and the framework does not need a global or central planner to assign tasks and coordinate the path planning. The agents can only share information with nearby agents when they are in close proximity to one another. They then use the newly received information to avoid conflict, such as collision while traveling and determine where printing needs to be done. The results of this decentralized planner are compared with that of a centralized multi-robot planner, which was presented in our prior work. That centralized planner uses a modified Genetic Algorithm to schedule and assigns the tasks to the individual agents and an A* path planning algorithm to obtain collision-free path planning for the generated schedule. The comparison is made based on criteria, such as scalability, computational time, and uncertainty (e.g., robot failure).

6.3 Motivating Example

In attempt to print two hexagons (shown in Figure 6.2) using two printing robots, we sliced a each of the hexagons individually using Cura slicer. These hexagons were then assigned to the printing robots. Since the geometries of the two hexagons are the same and were sliced using the same parameters, logic tells us that if robot one was to print hexagon

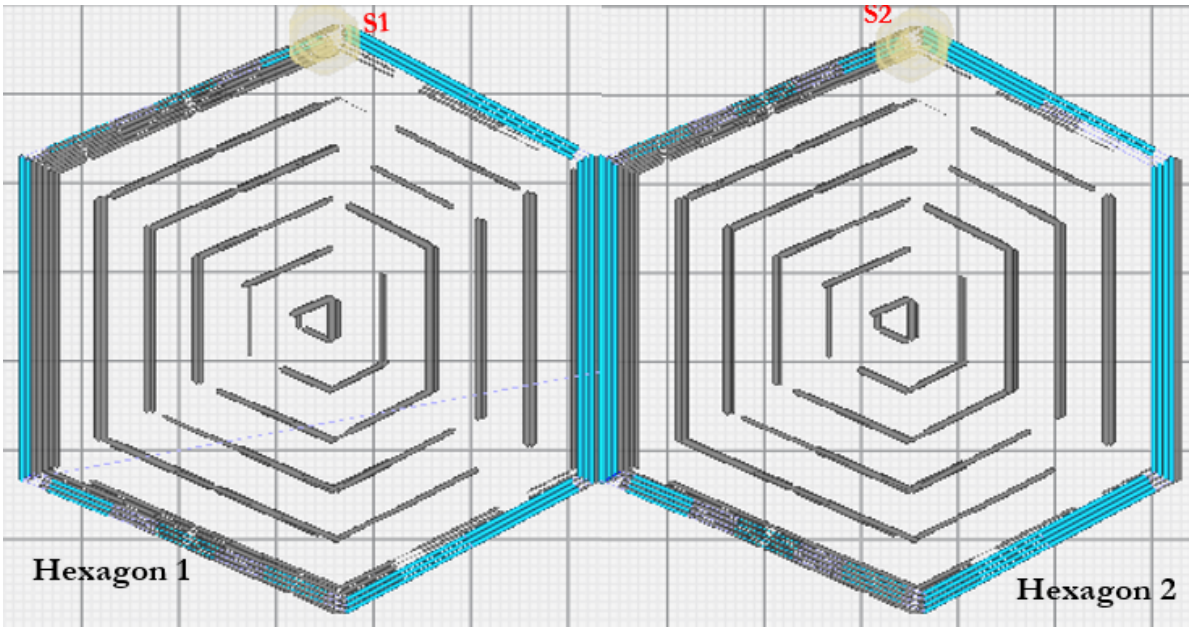

Figure 6.2: Two hexagons to be printed using two robots.

one with starting point highlighted in Figure 6.2 (S1) and robot two was to print hexagon two with starting point highlighted in Figure 6.2 (S2), the printing of the two hexagons should be executed synchronously, without the printheads of the printers interfering with one another. While theoretically, they should not collide, we found out during the implementation that was not the case. The two robots indeed did collide while printing the two hexagons. Not only that, since the printing was repeated multiple times to eliminate any systematic issues associated with the printer or the printing parameters, the collision between the printheads happened at different times or different location for each repetition. This demonstrated that even if we plan the entire manufacturing process taking every details into consideration, the implementation will be different from the plan. This demonstrates that the application of centralized approach, where planning is done prior to the implementation and communicated to the individual agent, in its current form could struggle to address the manufacturing prob-

lems due to inherent uncertainties in the manufacturing processes. The centralized approach has to be redefined so that it can better encapsulate such problems in manufacturing. This is what motivated us to explore decentralized planning approach for C3DP, that has potential to be more resilient to uncertainties that are rampant in manufacturing, especially additive manufacturing.

6.4 Literature Review

Multi-robot task planning has seen applications of both centralized and decentralized approaches. While more recently, an increasing number of decentralized and distributed approaches have been proposed, traditionally, the centralized approaches have dominated the multi-robot task planning problems.

6.4.1 Centralized Planning Approaches to Multi-Robot Planning

The use of a centralized approach for multi-robot planning is ubiquitous in literature. Though multi-robot planning includes multi-robot task allocation (MRTA) and path planning (MAPF) to undertake the allocated task, the two tasks are rarely studied together. This is because each of these tasks is an NP-hard problem [73, 56]. The centralized approaches to MRTA include optimization-based approaches. Darrah et al. also used MILP to solve the MRTA problem in the context of unmanned ariel vehicles (UAV) [74]. Large usage of the centralized approach is covered by metaheuristic approaches for MRTA. For example, Wei et al. used particle swarm optimization for cooperative multi-robot task allocation using a multi-objective (total team cost, balance of workloads) approach [47]. In another study, Sarkar et al. presented another heuristics approach called nearest-neighbor-based clustering and rout-

ing (nCAR) that scales better compared to other existing state-of-art heuristics ($O(n3)$) [75]. More recently, Zitouni et al. presented an approach using a two-stage methodology where at the global level, task allocation is done by using firefly algorithm, and local allocation is done by combining quantum genetic algorithm and artificial bee colony optimization [76]. In addition, some of the other heuristic approaches for MRTA include simulated annealing [42, 77], tabu search [77, 78], etc. On the other hand, MAPF has also been studied widely using different centralized approaches. Thabit et al. presented a multi-robot path planning approach based on multi-objective (shortness, safety, and smoothness) particle swarm optimization in an unknown environment [79]. Additionally, several heuristic approaches are inspired by the biological system, such as Genetic Algorithm [45], Ant Colony Optimization (ACO) [80], Particle Swarm Optimization (PSO) [47], and have been used to solve path planning problem. Other heuristics include the Simulated Annealing algorithm [81] and tabu search [82]. While such a heuristic approach provides good results, they have two limitations. First, it assumes prior knowledge of the environment, which might not be valid in a setting where the environment cannot be known beforehand (e.g., search and rescue disaster recovery). Second, the computational cost of the approach exponentially increases with the increasing scale of a problem. While approaches such as Rapidly Exploring Random Tree (RRT) have been proposed to solve path-planning in an environment that is dynamic and unknown beforehand [83], it still suffers from the curse of dimensionality of search space and does not work well with the geometric nature of the obstacles. Additionally, conflict-based search (CBS) algorithms solve the MAPF problem by breaking the search space into a large number of constrained single-agent pathfinding problems. This allows each of the problems to be solved in linear time, and the number of agents contributes exponentially to the length of

the final solution [84].

6.4.2 Decentralized Planning Approaches to Multi-Robot Planning

While the centralized approaches can produce an optimal or near-optimal solution for small-scale problems, they usually struggle in a non-deterministic environment. This is because everything in a centralized approach has to be pre-planned before implementation. While it is possible to enforce the synchronization of execution between multiple robots, the sequence of the execution is largely unsynchronized and non-deterministic. As the number of robots increases, it becomes increasingly difficult to predict the outcome of the planning over an extended period of time with a centralized approach. In addition, the communication cost will also scale non-linearly, which may result in difficulties with centralized approaches. In such scenarios, a more decentralized approach might make more sense due to their ability to work in uncertain or unpredictable environments and ability to work without a centralized planner. One widely publicized research using a decentralized approach is conducted by Werfel et al. where, the authors developed a swarm of termite-inspired multi-robot construction system that was solely based on a set of simple rules and local communication between the robots [9, 13]. The system consists of individual robots with very limited capability that can pick and place blocks for the construction of general structures, and the coordination between the individual robots was achieved by mimicking stigmergy. As a part of project TERMES, they developed both the hardware and software system and demonstrated the construction of large 3D structures. A reinforcement learning method was used to learn decentralized policies that seek to minimize the total construction time in the same system by Sartorettoi et al. [85]. While such an approach demonstrates speedup, it can limit the scalability of the

system. Similarly, Ortiz Jr. et al. presented the centibots system – a multi-robot distributed system consisting of more than 100 robots in unknown indoor environments over extended periods of time for search and rescue problems [86]. Peres et al. presented a multi-agent swarm robotics architecture to simulate heterogeneous robots that interact with each other and also humans to accomplish several types of missions [87].

While both the centralized and decentralized approaches in the current literature provide good solutions to the problems they address, none of the literature discussed above provides a good comparison between the centralized and decentralized approaches in the context of manufacturing using multiple mobile 3D printing robots. While there are some comparative studies between the centralized and decentralized approach [88, 89] in the context of the multi-robot system, the application domain is limited to discrete tasks such as pick and place, team formation, warehouse functionalities. The multi-robot C3DP poses more challenges compared to such discrete tasks due to the introduction of the manufacturing process. Thus, this study aims to address the gap due to lack of knowledge on how each of the methodology (centralized vs. decentralized) would perform in the C3DP application context by presenting a decentralized approach based on a simple set of rules and comparing the results with the centralized approach that uses modified GA with CBS.

6.5 Approaches to C3DP Planning

6.5.1 Rules-Based Approach to C3DP

In this decentralized approach, each mobile 3D printing robot is an autonomous agent and can make decisions based on local information. There is no central planner that assigns

the printing tasks to individual robots and schedules them to move to specified locations for printing those assigned tasks. Instead, the agents adhere to a set of simple rules and make decisions according based on the said rules. We outline these rules in more detail in the subsequent sections. In the subsequent discussion, a job refers to the entire printed object, which is divided into small chunks, and a chunk is a subset of that job. A job is split into chunks so robots can print portions of the job, and as the portion of the job gets printed by individual robots, eventually, the whole job will be completed. An example of chunking is presented in Figure 6.6, where a rectangular bar is chunked into thirty chunks. Each of these chunks has different shapes and sizes and would result in different print times. In this approach, robots are equipped with the following capabilities:

1. Robots can move freely from one grid point to the next in any cardinal direction on a grid floor.

2. Robots can only print in two directions, up and down.

3. Robots have a limited communication range, i.e., they can communicate with surrounding robots within a two-grid-point radius.

4. Robots can read coordinate information from grid points when they move to a grid point.

5. Robots are enabled with close-range sensors to view their local surroundings for obstacles.

6. Prior to printing, robots are loaded with job information, including:

(a) G-code of all chunks in the job, which can be accessed when they are ready to print individual chunks.

(b) The location of each chunk to be printed.

These capabilities allow robots to act independently and access sufficient information to decide on whether they can print at a certain location. In this approach, the job is chunked and placed on the floor beforehand. This information is passed to the robot prior to printing. Robots have five states: searching, printing, orbiting, charging, and reloading. They first start with the searching state, where robots move towards the center area of the grid, where the job is located using the coordinates information of chunks. Once a robot moves from one grid point to another grid point, it searches for the new grid point in a lookup table. If the grid point returns a match, it means a chunk is to be printed at the grid point. If such location is found, it determines whether it is allowed to print at the location using the Geometric Rule, discussed in Section 3.1.1. If it can print, it will transition to a printing state and start printing the chunk. On the other hand, while searching, once a robot finds a chunk printed (or is currently printing) by another robot, it transitions from the searching state (looking for where the job is located in the entire floor space) to an orbiting state, where it starts to orbit the printed chunks and the printing robots. Robots switch between the orbiting state and the printing state until the entire job is finished.

As the robots print the chunks and travel from one location to another, they dissipate energy and might need recharging at some point during the execution of a job. Thus, if the battery level of the robots gets below a pre-determined threshold (e.g., 80 %), they will need to be recharged and, therefore, will move toward the charging station located at one of

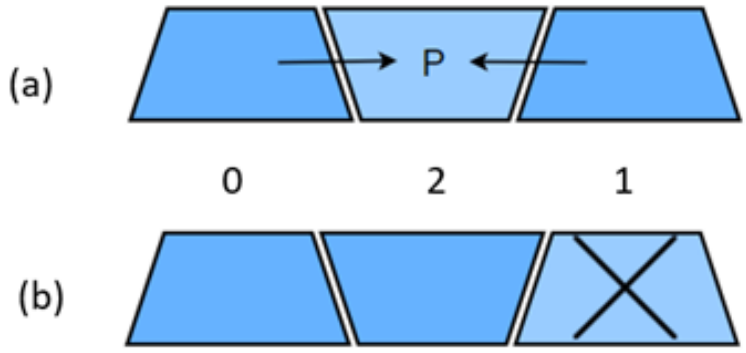

Figure 6.3: (a) Rule of geometry indicates that chunk 0 and chunk 1 should be printed before chunk 2 (b) Possible issues resulting from the lack of geometric rule where, chunk 1 cannot be printed if chunk 2 is printed first.

the corners of the floor. Once the robots reach the charging station, they transition to the charging state. The batteries of the robots must be fully recharged before they can leave the charging station. Once fully recharged, they will transition back to the searching state and search for a print location. Similar to the charging state, robots might run out of printing filament during a large print job. If that happens, it will transition to the reloading state and notify the user to reload the filament. Once the new filament is loaded, it transitions back to the state before running out of filament. However, only three states: searching, orbiting and printing state are considered in this study. Charginig and reloading states are not taken into consideration.

1. The Set of Rules

   (a) The rule of geometry

   In Chapter 3, we briefly presented a sloped-surface chunking strategy where a part is divided into smaller chunks such that each chunk has sloped surfaces on all four sides (except for the end chunks). This sloped surface chunking allows

adjacent chunks to be printed on the sloped surface of the already printed chunk, creating an instant bond between the two adjacent chunks. Since the adjacent chunks have a sloped surface interface with each other, this creates geometric constraints. The rule of geometry is established to ensure geometric dependencies between the adjacent chunks are followed. For example, as shown in Figure 6.3, it can be seen that chunks zero and one must be printed before printing chunk two because chunk two has overhangs that prevent parts of chunks zero and one from being printed. The printing nozzle of the robot will collide with the said overhangs of already printed chunk two while attempting to print portions of chunk zero and chunk one. To avoid such collisions, it is necessary that robots print chunks that have convex slopes (e.g., chunks zero and one) before printing adjacent chunks that have concave slopes (e.g., chunk two). Such geometric dependencies are embedded as directed graph data structures, where each node represents a chunk, and an edge between the node represents the dependency relationship between the chunks. Such data is provided at the beginning of the print job as part of initial global information. Thus, the rule of geometry is necessary for the sloped-surface chunking strategy to be implemented properly. Once a printing robot reaches a print location, it will search for information on whether the chunk at that location is concave or convex. Afterward, the robot can inspect the surroundings and determine whether a chunk can be printed at that location. For example, in Figure 6.3, if the robot reaches the print location where chunk two is to be printed, it will inspect whether chunks zero and one have been already printed or not. If either of the chunks zero or one has not been printed, chunk two cannot be

printed, and the robot will have to search for a different chunk to print. This rule is important as this allows for the print job to expand properly and not print chunks that would prevent robots from accessing other chunks during the printing. While the provided example uses a sloped-surface chunking strategy for demonstration, such dependencies will very likely exist in any other future geometric partitioning strategies. Thus, the rule of geometry has a general implication to handle such dependencies.

(b) The rule of intersection

The Rule of Intersection is used to help avoid a robot-to-robot collision. Since robots use local communication, there must be a way for robots to avoid colliding with each other if two or more robots are about to operate in an intersection at the same time. When robots are within each other's communication range, they will transmit their next move, and if both the robots want to move to the same location, a tie must be broken. To break a tie, both the robots generate a random number. The robot with the larger number will have the right of way, and the other robot will either wait until the first robot has made a move or move to a different location.

(c) The rule of orbiting

The Rule of orbiting is designed to limit random movements and bring a more systematic approach to robots' search for a print location. For example, when a robot is in searching, it can start randomly moving around the floor until it finds a print location, but such an approach could result in excessive movements.

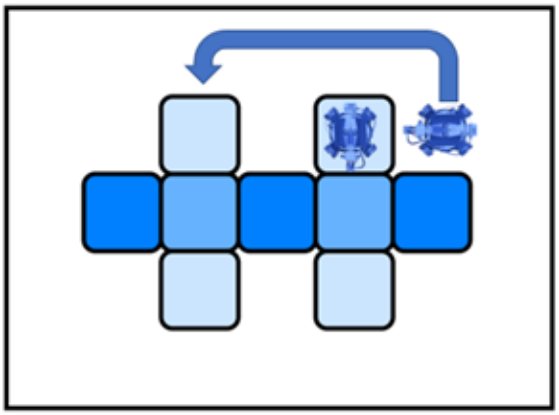

Figure 6.4: Rule of orbiting indicates that robot needs to start orbiting already printed chunk in counterclockwise manner to find new chunk for printing.

A robot could make hundreds of movements before finding the location to print. To avoid those movements, the robot will, in the beginning, move towards the center of the floor (where seed chunks are located). Once any one of the print locations is found, the robot can initiate printing. After the print is complete, the robot will start to orbit the printed chunks counterclockwise to search for the next print location. This approach is presented in Figure 6.4. Any robot is not allowed to turn back or circle the printed chunks in clockwise movements. This rule is inspired by the work done by Nagpal et al. [14].

6.5.2 The Centralized Approach for C3DP

In Chapter 5, we presented a meta-heuristic-based centralized approach to multi-robot scheduling based on a modified genetic algorithm (MGA). The genetic algorithm (GA) is an evolutionary stochastic algorithm that has widely been used in MRS problems as it provides satisfactory solutions to combinatorial problems. In the presented approach, the MGA randomly generates a population of initial chunk assignments and uses the dependency

list in conjunction with the chunk assignment to generate print schedules. Genetic operators are then applied to modify the chunk assignment until a specified number of population generations was achieved. The fitness function in the MGA was to minimize the total print time. While the previous approach was able to yield a near-optimal solution for a small-scale problem with 20 chunks and a large-scale problem with 200 chunks, it did not account for the travel time while the robots move from one print location to another. Using MGA for manufacturing scheduling would result in a shorter print time (due to better print schedules), but the generated schedule might result in longer travel time because the printing robots have to spend longer time traveling from one print location to another. Thus, to address this limitation, we improved the MGA in this study by taking path planning into consideration of the fitness function formulation.

To incorporates collision-free path planning in our previously developed MGA, a conflict-based search (CBS) method is adopted. CBS uses a two-level approach where the high-level search for collision-free path planning is done on a constraint tree. In such a constraint tree, each node specifies a time and location constraint for an agent. For each of such nodes, a low-level search is done to find paths for all agents that satisfy the node constraints. While CBS guarantees optimality by exploring all possible ways of resolving conflicts, it also can suffer from longer runtime if poor choices are made for conflicts to split on. The detailed implementation of CBS for multi-robot cooperative 3D printing is presented in our previous work [62]. Once the initial population representing print schedule is generated in MGA, path planning using CBS is done for each individual chromosome. The travel time obtained using CBS is then added to the fitness value of each chromosome using equation (6.1). Thus, the chromosome's overall fitness score includes the total print time

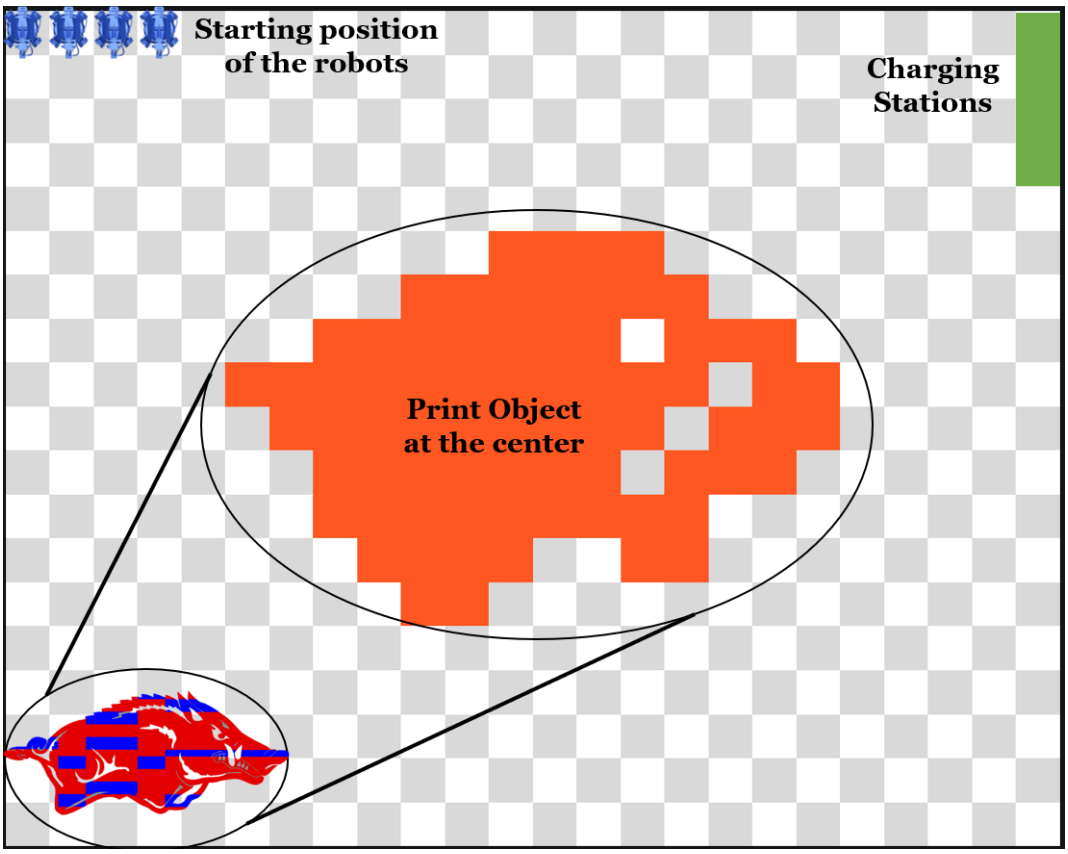

Figure 6.5: Representation of grid world where four robots are ready to print a job placed in the center of the grid.

and the total travel time required for a particular print schedule. This new MGA method enables the generation of solutions to improve both task scheduling and robot path planning simultaneously. Doing so, however, increases the overall runtime of the algorithm, as an instance of CBS has to be carried out for each chromosome in each iteration.

$$Total\ time = Max(T_{start,ij} + T_{print,ij} + \sum T_{travel,j}) \tag{6.1}$$

where, $T_{start,ij}$ is start time of chunk i on robot j

$T_{print,ij}$ is print time of chunk i on robot j

$\sum T_{travel,j}$ is total travel time of robot j throughout the job

$j = 1, 2, 3. \ldots\ldots m$,robots used for printing

$i = 1, 2, \ldots\ldots n$,chunks assigned to robot j

6.6 Results and Discussion

This section presents the experiment setup, the underlying assumptions, and the evaluation metrics used to compare the two approaches.

6.6.1 Experimental Setup

All the experiments for the comparison are conducted in Microsoft Window OS, and both approaches are programmed using Python programming language. Following assumptions were made during the implementation:

1. A job is placed at the center of the grid world with five extra grid points around it on all sides. Five grid points were left on all sides because that would provide enough space for multiple robots to move across simultaneously if needed. A representative visualization is presented in Figure 6.5. The coarse representation is used in order to minimize memory requirement for simulation and is only used for visualization. High resolution STL models could be used instead, which will improve the visualization but requires more memory.

2. All the robots, in the beginning, are placed around the origin, as shown in Figure 6.5.

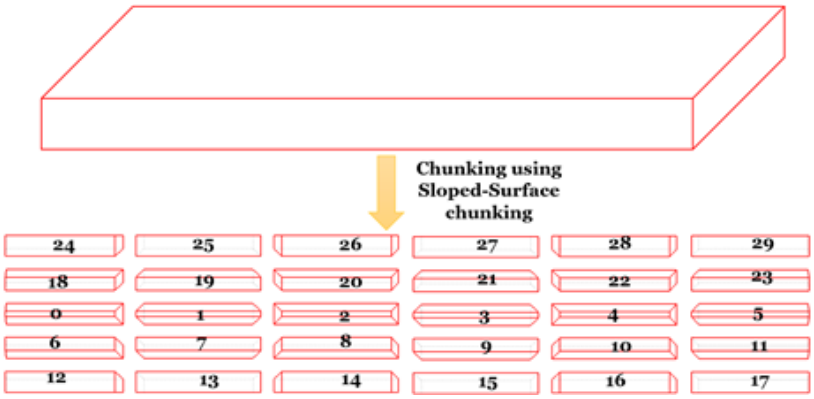

Figure 6.6: Entire rectangular bar and exploded view of chunks from top with marked chunk number.

6.6.2 Evaluation Metrics

To quantitatively compare the two approaches, we use the makespan of a job as an evaluation metric. Makespan includes is the time it takes the robot to print the job, . This includinges both the print time as well as and the travel time. To quantitatively compare the two approaches, we use the makespan of a job as an evaluation metric. Makespan is the time it takes the robot to print the job, including both the print time and the travel time. The computational complexity of the multi-robot planning increases with the scale of the problem, such as the number of robots and the number of chunks to be printed. Thus, we want to test each approach's performance when the scale of the problem increases. To this end, we apply both the rules-based approach and the MGA with CBS in two different case studies. In the first case study, a simple geometry is divided into multiple chunks and assigned to the robots for printing. In this case study, the number of resulting chunks is varied, ranging from 20 chunks (small part) to 600 chunks (large part). This allows us to see how the performance (e.g., makespan and computational time) of the two approaches change with the increase of the number of subtasks. The second case study uses a more

complicated geometry (a razorback mask). With a more complicated geometry, when a part is chunked, the size and shape of the chunks vary widely. Such differences in the shape and size of the chunk is highlighted by marking few chunks in blue in Figure 6.5. The printing of chunks in the case of a rectangular bar can be highly synchronized. For example, chunk one and chunk three could be printed by robot one and robot two in Figure 6.6. Since they have similar shapes and sizes, they will be completed together (assuming the robots have the same printing parameters and perform printing tasks at the same speed). Once the printing is complete, robot one and robot two could move to the right and start printing chunk two and chunk four. The same strategy can be applied to print every row. Thus, the chunks can be printed by the robots together, and they can move together, which would result in a much shorter makespan. However, for the razorback mask, none of the chunks have similar geometries and sizes, and thus, such synchronicity cannot be achieved. We want to see how these two approaches perform in such situations. Thus, the main purpose of the second case study is to test whether the aforementioned complexity in terms of wide variation in shape and size (and thus, the print time) of geometry has any impact on the performance of the two approaches. In addition to this, we also conduct additional simulations to test the ability of both approaches in handling uncertainties during the actual 3D printing, which could cause a variation between the estimated print time and the actual print time. The result details regarding each case are presented in Section 4.3. The assumptions adopted in the case studies are summarized as follows:

1. All robots are homogeneous. That means every robot uses the same parameter settings and print settings and thus, spends the same amount of time traveling from one grid

point to another as well as printing the same chunk.

2. In order to make the calculation more realistic and match the actual printing scenario, an object is chunked first using chunker, demonstrated in our previous study [58, 90]. The STL model of the individual chunks is then passed through a slicing engine such as Cura to get a better estimate of the print time. The print time obtained from the slicer is used to calculate the simulation time (to simulate printing in rules-based approach) using equation 6.2.

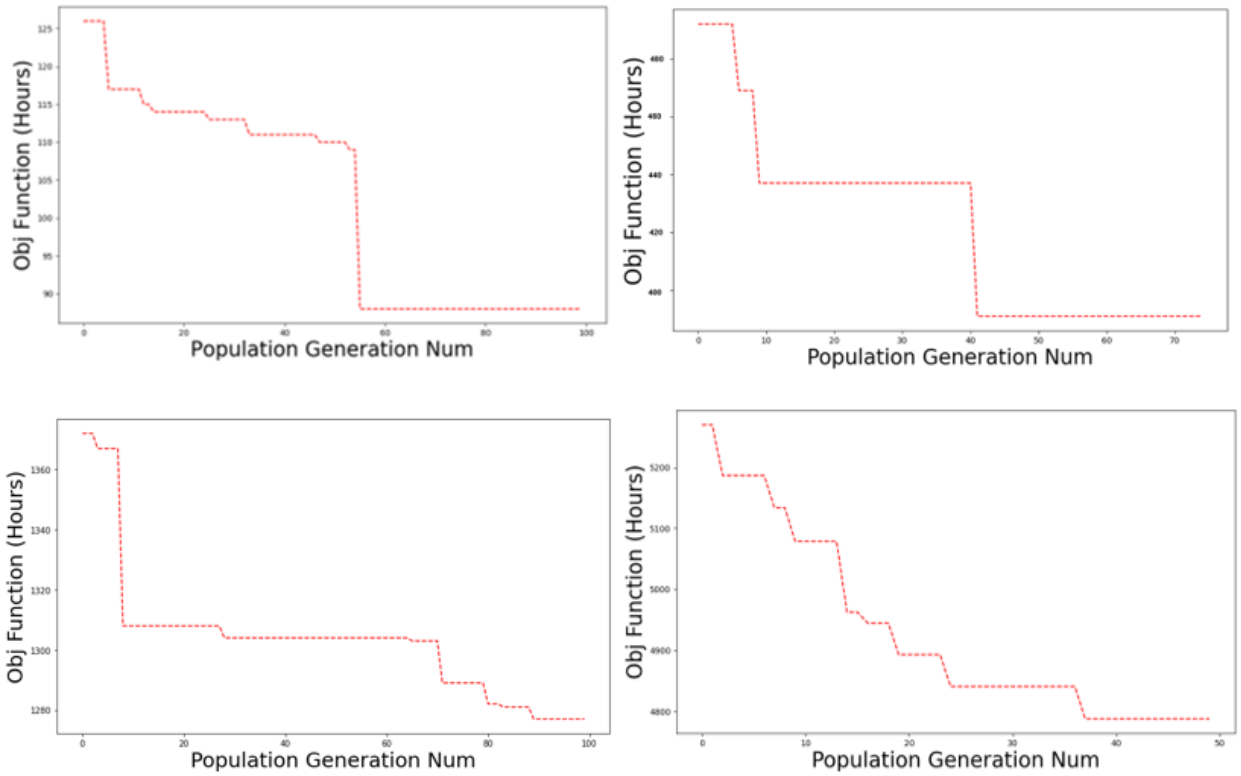

Figure 6.7: Graph showing the change in objective value for centralized approach for different scale of problem.

$$Simulation\ time = \frac{\sqrt{\frac{Print\ time}{10}}}{6}\ timesteps \tag{6.2}$$

3. The location of the job is determined beforehand by the user and is not part of planning for multi-robot C3DP.

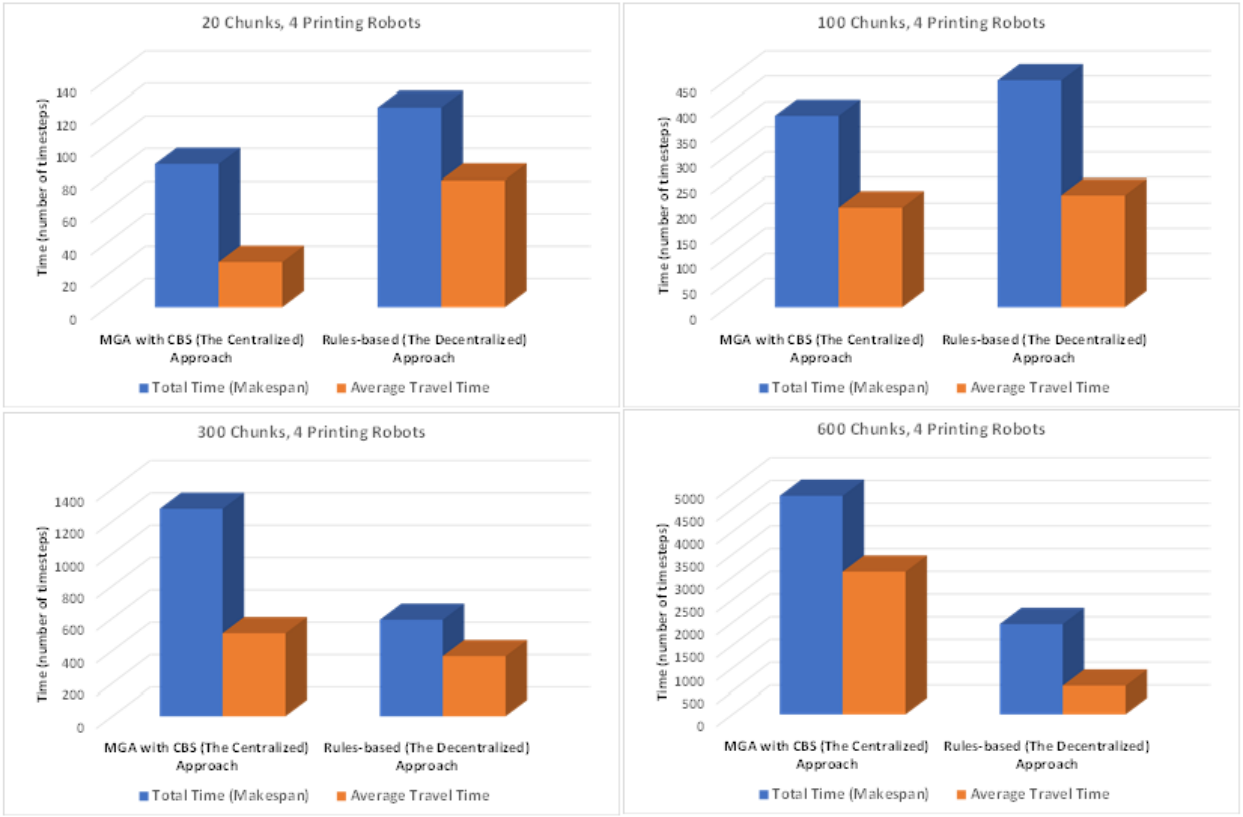

Figure 6.8: The makespan and average travel time by robots in both decentralized and centralized approach for different scale of problems in case study I.

6.6.3 Case Studies

Case I

The first case study is a rectangular block of dimension $1m \times 0.8m \times 0.015m$ and has a total volume of $0.012m^3$. The rectangular block and the resulting chunks using the sloped surface chunking strategy are presented in Figure 6.6. Four robots are used to print these chunks. In order to better understand how the behavior of both approaches changes with

Table 6.1: Metrics associated with case study I

	Sm. Job (20 chunks)		**Med. Job (100 chunks)**		**Lg. Job (300 chunks)**		**Ex-Lg. Job (600 chunks)**	
	Dec.	Cen.	Dec.	Cen.	Dec.	Cen.	Dec.	Cen.
Makespan	123	88	448	379	597	1277	1964	4788
Avg. Travel Time by an Agent	78.25	28	222.25	198	365.25	507	609.75	3118
Min. Travel Time by an Agent	68	23	203	178	350	436	583	3633
Max. Travel Time by an Agent	98	36	232	212	393	670	638	2596
Max. # of Chunks Printed by an Agent	6	5	27	28	79	77	153	167
Min. # of Chunks Printed by an Agent	3	5	24	19	68	71	147	139
Increase in runtime	-	-	-	7X	-	28X	-	39X

the scale of the problem, we change the scale of the object and calculate the result using both approaches. First, the scale of the object is increased by five times, which resulted in 100 total chunks. To see how the trend changes with the scale, we further increased the scale of the problem by 15 times (resulting in 300 chunks) and finally by 30 times (resulting in 600 chunks).

The results obtained using both the centralized and decentralized approaches are presented in Table 6.2. The information presented includes the total makespan, which is the sum of printing time and traveling time. It also reports the maximum traveling time for any robot, i.e., the longest path a robot travels throughout the entire print cycle of the job. In addition, the table includes the least amount of time that a robot spends traveling.

The maximum and the minimum number of chunks printed by any robot are also reported. Finally, the overall change in the objective function with respect to the change in iteration (population generation) is presented in Figure 6.7.

Case II

The second case study is a razorback mask, which has a more complicated geometry than the rectangular bar presented in the first case study, in terms of shape and size variation. A coarse representation of the razorback mask is presented in Figure 6.5 and has the dimension of $610mm \times 267mm \times 52mm$. The razorback mask was chunked into 50 chunks and was printed using four printing robots.

6.6.4 Discussion

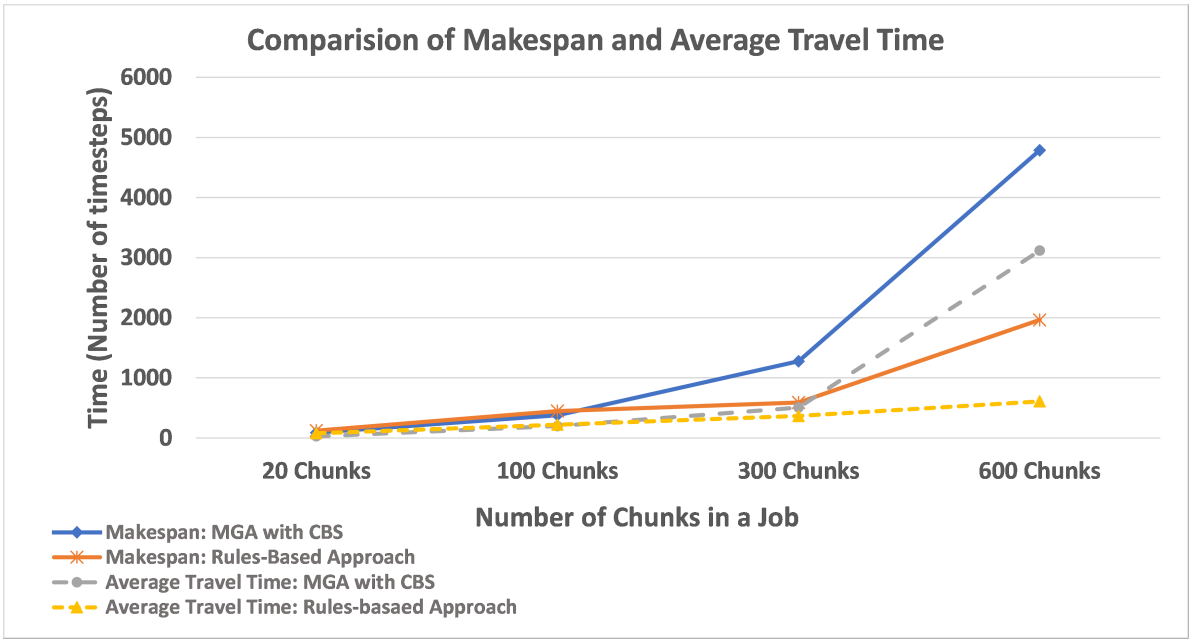

Figure 6.9: The changes in makespan and average travel time obtained using different approach for different scale of a problem presented for case I.

Table 6.2: Metrics associated with case study II

	Rules-based Approach	**MGA with CBS**
Makespan	184	167
Avg. Travel Time by an Agent	124.25	107
Min. Travel Time by an Agent	113	95
Max. Travel Time by an Agent	144	110
Max. # of Chunks Printed by an Agent	13	15
Min. # of Chunks Printed by an Agent	12	11

The charts presented in Figure 6.8 illustrate the makespan and average travel time of both approaches in Case I. As we can see in the figure, the makespan for small-sized as well as medium-sized jobs in Case I for using the centralized approach is shorter than that using the decentralized approach. However, the trend changes as the size of the job go up from medium to large, which can be observed in the graphs for 300 chunks and 600 chunks. The change in the trend can be observed in the graph presented in Figure 6.9 as well. This is likely because as the number of chunks increases in the job, the search space increases significantly, which results in the centralized approach not being able to search the entire search area in a reasonable amount of time. This diminishes the likelihood of a centralized approach being able to find optimal or near-optimal solutions to these problems. In addition to this, as the size of the problem increases, the computational time for the centralized approach increases significantly as well. While the computational times for the small and medium jobs were in minutes ($<$ 20 minutes), the ones for the large job and extra-large job were in hours ($\sim$ 13 *hours*). Thus, to limit the computation time, only 50 iterations of the centralized algorithm were allowed to run for the extra large job. While putting

such a limitation has impacted its ability to obtain a better result because the centralized planner could not converge towards the best solution, it demonstrates the computational challenges of the centralized approach and highlights its high demand for computational resources. Even with a limited number of iterations, it took the centralized approach almost 13 hours to run the extra-large-sized job. This marks an increase of over 39× compared to the small-sized job with 20 chunks. Additionally, the computational time for the medium-sized job increased by 7× (134 *minutes*) and that of the large-sized job increased by 28× (8 *hours*) compared to the small-sized job,. In order to reduce the computation runtime, a print schedule could be generated first using the MGA method without integrating path planning, followed by the implementation of CBS on the generated schedule to obtain a collision-free path. This, however, could lead to a longer travel time as such a schedule generation does not consider path planning. But in the light that robots spend 90% of the time printing while 10% for traveling (e.g., in cases where chunks are large and printing a single chunk could take hours, whereas traveling from one grid point to another only takes a few seconds), this approach might yield acceptable solutions. Thus, the synchronous scheduling and path planning used in the centralized approach yields a better result, where both the schedule and the path planning are computed concurrently; it also demands more computational resources. This is especially true for extremely large-scale problems consisting of a large number of chunks and robots, as demonstrated by the large and extra-large jobs. Meanwhile, the computation time of the decentralized approach did not change much for different scales of the problem. This is because no real computation is required for planning. As robots start from their initial position, they make movements based on these rules and make decisions instantaneously without planning for any future events. Not having a need

to plan for the future gives the robot flexibility and freedom from the computational burden. Although looking at the graph and the data associated with case study I, one can come to a conclusion that the centralized planner is always worse than the decentralized planner for a large-scale problem in terms of the overall makespan,however, it is not always the case. Given enough computational resources and time to converge to a solution, the centralized planning approach can outperform the the decentralized planning, even for a large scale problem.

The data associated with both approaches for the second case study are presented in Table 6.1. The makespan of both approaches, while not exactly equal, is similar. The results obtained using the centralized approach are marginally better than the one obtained using the decentralized approach both in terms of makespan and the average travel time. This shows that the geometric complexity of the part does not have a significant impact on the performance of either of the two approaches.

Uncertainties due to variable print times

An accurate estimate of a print time for a digital model is essential as it impacts the overall scheduling, the availability of robots for printing, and the overall cost estimation. While the slicing engines, such as Cura, Slic3r, Simplify3D, Repetier, and OctoPrint, provide a reasonably good estimate of print time, the actual print time during implementation can often generate a discrepancy. And, depending on the size and complexity of the part to be printed, the discrepancy could range from minutes to hours. For example, the time estimation of a part with relatively simpler geometry (e.g., the entire rectangular bar in Figure 6.6) will be close to the actual print time. On the other hand, the time estimation of a more

complicated geometry (e.g., the Razorback in Figure 6.5) will have a higher discrepancy with the actual print time. Such discrepancies can be attributed to a combination of uncertainties due to geometric attributes (e.g., dimension, number of contours, etc.) and hardware differences (e.g., set value of print speed vs. actual speed). Moreover, the discrepancy can vary from one print to another, even for the same part. Although such discrepancies generated from one robot might not be significant, the problem becomes prominent as hundreds or thousands of robots work together due to propagation or cascading effects.

Therefore, we conducted a Monte Carlo simulation to evaluate how each individual planning approach handles the uncertainties. By literature review, we find that the extant literature lacks studies on the discrepancies between the estimated and the actual print time in the FDM process. Thus, we chose Gaussian distribution to simulate the variation in the actual print time. Once a part was divided into multiple chunks, each chunk was sliced using a Cura slicer engine from which an estimated print time will be obtained. This print time was used as the mean value to generate the distribution. Since the geometry of chunks and the scale or the size of the part plays a vital role in the resulting discrepancy, it makes sense to choose the standard deviation, that is some factor of mean value. However, due to the lack of proper guides in the existing literature on choosing the value for standard deviation for variation in print time, one-tenth of the mean print time was used for it. A printing scenario with 60 chunks using four printing robots was used to demonstrate the simulation. In the case of the decentralized approach, the estimated time for the print job was 133 timesteps, but as the histogram in Figure 6.10 shows, the overall result could range from 123-152 timesteps. Such variation in print time does not seem to impact the overall performance of the rules-based approach, as the schedule is not predetermined. The chunks

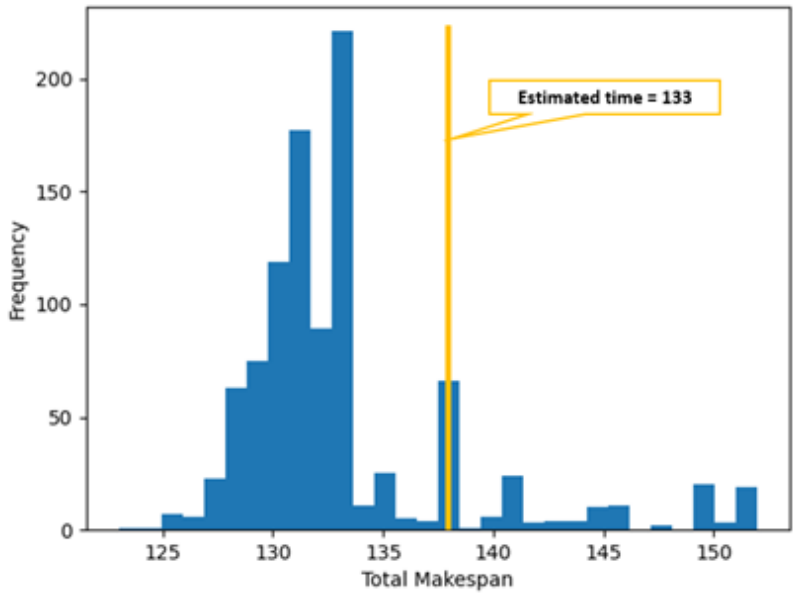

Figure 6.10: Total makespan obtained after running Monte Carlo simulation that shows variation in actual print time in the decentralized approach. Estimated print time marked for reference.

are printed as the printing robots discover them. Thus, the rules-based approach remains largely unaffected by uncertainties. The centralized approach using MGA and CBS, on the other hand, generates the print schedule as well as the path planning before the actual print takes place. Therefore, in the same printing scenario for the centralized approach, it takes 124 timesteps. However, due to the variation of the actual print time, we observe a larger range for the centralized approach, ranging from 100-275 timesteps, as shown in Figure 6.11. This is because, to avoid potential collisions between printing robots or between a robot and printed material caused by the discrepancies, the print of the next batch of chunks cannot be started until all of their dependent chunks are printed. For example, a schedule might result in a scenario where robot 1 is planned to print chunk 8 first and then chunk 9 in Figure 6.6. Similarly, robot 2 is planned to print chunk 10, followed by chunk 11. Let us assume that each chunk is estimated to be printed in 50 minutes, but due to discrepancies in the actual

print time, robot 1 might finish printing chunk 8 in 60 minutes, whereas robot 2 might take 70 minutes to print chunk 11. After 60 minutes (once chunk 8 is printed), robot 1 is scheduled to move to the right to start printing chunk 9. However, robot 2 is still printing chunk 10 adjacent to chunk 9. This results in a collision between the two robots and eventually print failure. Thus, a newly added constraint that forces robot 1 to wait for additional 10 minutes before moving to the right to print chunk 9 causes the overall delay for the entire print. While such collision avoidance is already integrated into the schedule generated by the central planner, collision-free printing is only guaranteed if the generated schedule is implemented using the estimated time. However, with the actual printing widely varying from the estimated printing times, such a collision-free printing cannot be guaranteed. Thus, additional precautions are required to prevent accidents, and the optimal or near-optimal schedule that was generated during planning is no longer valid during implementation. This makes the use of the optimization approach inconsequential as optimal planning no longer guarantees optimal performance. Thus, the centralized approach suffers as a consequence of uncertainties in the printing time.

While the centralized approach could suffer as the consequence of such uncertainties, it is also important to highlight that the centralized planning, in conventional sense, is not designed to address such problems. A centralized planner that has a capacity to handle real-time feedback could handle such uncertainties better but at the same time would result in higher operational cost. Thus, to effectively apply such a centralized approach in the manufacturing environment with a lot of uncertainties, a problem reformulation could yield better result. However, it requires further deliberation and is a part of future research.

Thus, the results show that, for smaller cases, the centralized approach outperforms

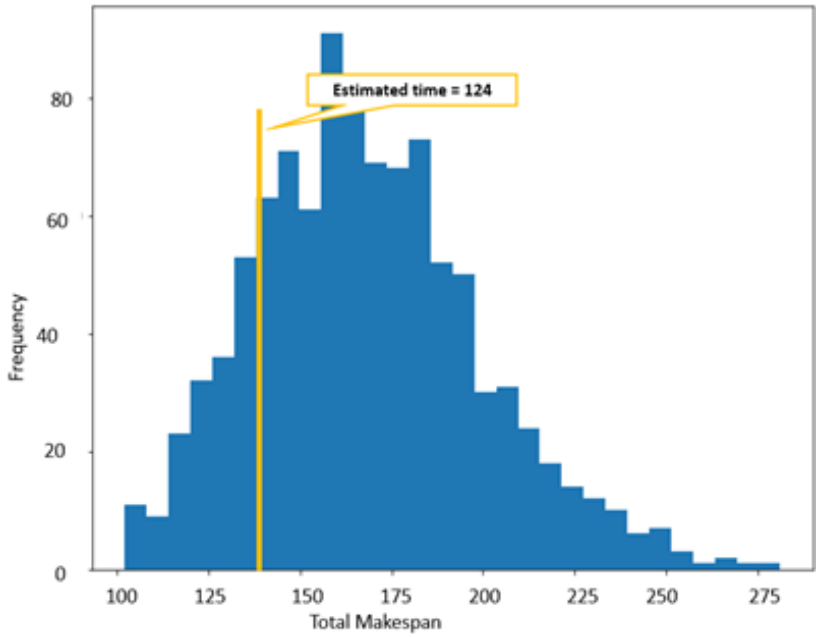

Figure 6.11: Total makespan obtained after running Monte Carlo simulation that shows variation in actual print time in the centralized approach. Estimated print time marked for reference.

the decentralized approach in several metrics, including the total makespan and average travel time of a robot. However, the trend reverses for the larger scaled problem. The centralized approach has ability to outperform the decentralized approach, even for large-scale problems, but that requires much higher computational resources and longer runtime to converge to a solution. In addition to this, uncertainty would have no adverse effect on the decentralized approach, whereas such uncertainties could result in the chosen plan being inaccurate and could potentially lead to accidents and potential failure. The summary of these differences based on the result and discussion is presented in Table 6.3.

Table 6.3: Summary of the centralized and decentralized approaches in C3DP context

Centralized Approach	Decentralized Approach
Shorter makespan, shorter average travel time for smaller job	Longer makespan, longer average travel time even for smaller sized jobs
Large computation time for larger job	Computationally not as time consuming
Requires longer time to converge to near optimal solution for large-scale problems	Quality of solution not affected by size of the job
Affected adversely by uncertainties in printing time	Unaffected by the uncertainties in printing time

6.7 Conclusion

In this chapter, a rules-based decentralized approach and an MGA-based centralized approach are introduced. The centralized approach is an extension of Chapter 6, but further integrates CBS-based path planning in generating schedules. For both approaches, the job is chunked, and floor space is allocated prior to printing. The rule-based decentralized approach allows robots to make independent decisions based on their local surroundings and job information. The centralized approach, on the other hand, plans the print scheduling, assignment, and path from one print location to another and outputs a full schedule based on the MGA using CBS.

Two case studies are presented to compare the performance of the centralized and decentralized approaches and analyze their advantages and disadvantages in the C3DP context. The first case study was a simple geometry (a rectangular bar) consisting of a varying number of chunks, ranging from 20 chunks to 600 chunks. The primary aim of the first case study was to check the scalability of the two approaches to understand how the result (makespan), as well as the runtime of the algorithm, changed with the increase in the size of the problem. While the centralized approach outperformed the decentralized approach for

the small-sized and medium-sized jobs, the trend reversed for the large and extra-large-sized problem. The second case study is a large-scale razorback mask consisting of 50 chunks and is to be printed with four robots. In Case II, the makespan of both jobs was similar, with the rules-based approach yielding a slightly higher makespan. This leads us to believe that the geometric complexity might not have as big of an impact on the overall performance of the two approaches. A notable issue that the centralized approach runs into is the increase in computation time for planning the larger jobs. This affects the scalability and robustness of the centralized approach. This is an advantage offered by the decentralized approach. In addition to this, the decentralized approach also handles the uncertainty in the printing time very well compared to the centralized approach. An uncertainty in print time, a common occurrence in 3D printing, could result in catastrophic failure in a centralized approach, whereas it does not have an impact on the functionality of the decentralized approach. Similarly, if a robot fails while traveling, other robots will work, continues working, and take a larger chunk of work to complete the job. A centralized approach, on the other hand, requires replanning of the job.

An exciting future work could involve further generalizing the rule-based decentralized approach to work with alternative chunking strategies. In addition to this, a more robust approach could be used that would minimize the total travel of the robots in a decentralized approach. However, doing so might require a more computationally taxing algorithm. And, finally, as the centralized approach is more likely to run into problems with the size of the job, a hybrid approach could provide a good solution, that has the characteristics of both the decentralized (to handle uncertainty) and centralized approach (to achieve optimal results) in hopes that similar performance is achieved but with better scalability and increased

robustness. In addition to this, a reformulation of the centralized approach that could handle uncertainties in manufacturing could serve the manufacturing community and help the centralized approach achieve better results in presence of such uncertainties.

7 MECHANICAL STRENGTH OF PARTS PRINTED USING CHUNK-BASED COOPERATIVE 3D PRINTING

7.1 Overview

On the most fundamental level, the task division in C3DP is to partition the geometry of the given object to be printed using different mobile 3D printers. Geometry partitioning has been extensively studied in the fields of computer graphics and computer-aided design (CAD) under the topics of mesh partitioning, volume decomposition, and image segmentation [91, 92]. Many methods are developed for automatic partition of the models, such as binary space partitioning [93], where models are partitioned into two convex shapes recursively using hyperplanes, normalized cuts [94], where cut cost is calculated as a fraction of the total edge connections to all the nodes in the graph, and region growing [95], where seed points are selected based on preset criteria (such as pixel intensities) and the regions are grown from these seed points to adjacent points depending on the said criteria. These methods can be generally put into two categories: top-down and bottom-up. In the top-down approach, an original model is cut into smaller pieces recursively based on heuristics. The stopping criteria could be the desired volume [96] or the number of sub-parts [89]. The bottom-up approach typically takes two steps: seeding and clustering [94]. For seeding, some primitives are selected as starting points, which are then clustered or merged into sub-parts based on different affinity measures [94]. The boundary between sub-parts generally takes three different forms [97]: plane [94], voxel [98], and free-from [58]. While these methods can be used to divide a large object into smaller parts to be assigned to the printers for

printing, they do not consider the dynamic constraints necessary for enabling C3DP without post-processing needs such as labor-intensive gluing of chunks following the printing process. Thus, even though in fundamental level geometric partitioning is similar to chunking, chunking requires additional considerations that are not considered in static portioning of the geometry, seen in geometric partitioning research. So, while printing a larger part using multiple robots in a simultaneous manner usually requires that the part be partitioned somehow into smaller subparts, the manner in which the part is partitioned affects the number of robots that can be employed simultaneously to print the part as well as the overall integrity of the part. Thus, the chunking distinguishes itself from general static geometric partitioning problem in three major areas:

1. It needs to ensure that the printer can continue to print the side of printed chunks without colliding with the already printed chunks.

2. It should allow printing multiple chunks simultaneously manner.

3. The printed chunks should be bonded together while printing without compromising the strength of the printed part.

To address these three requirements, we used a sloped plane to divide an object into chunks, as illustrated in Figure 7.1. This chunking strategy is called the sloped-surface chunking strategy. The sloped-surface chunking strategy allows the cooperation of robots (i.e., simultaneous printing or parallel printing) at the chunk level.

The chapter is organized as follows:

7.2 Introduction to sloped-surface chunking strategy introduces the concept of sloped-surface chunking strategy and defines some of the parameters that are considered,

while chunking using the sloped surface.

7.3 Mechanical Strength of Chunk-based Parts: Introduction and Background identifies some of the parameters that could impact the mechanical strength of chunk-based parts and discusses some of the relevant literature in the field.

7.3 Experimental Setup and Methodology outlines the details of the experimental setup and tools used to undertake such experiments as well as the methodology adopted to undertake experiments.

7.3 Results and Discussion presents the results obtained from the experiment and analyzes the correlation of the identified parameters with the tensile strength of the chunk-based parts.

7.4 Conclusion provides concluding remarks and important lessons learned from the experiment and analysis.

7.2 Background of Sloped-Surface Chunking Strategy

In the sloped-surface chunking strategy, a CAD model is divided into multiple chunks using alternating chunking angles so that there is a sloped interface between the adjacent chunks. In the sloped interface chunking, a part is divided first along one axis (either X or Y) and then along another axis (either Y or X) such that each of these chunks has either a positive or negative slope on each side, as shown in Figure 7.1. The positive slope refers to a slope with an angle θ_c smaller than 90°, whereas the negative slope refers to the one with an angle larger than 90°. For example, the chunk in Figure 7.1(d) shows a central chunk with all four positive slopes. The shape of the chunk differs based on the sloped surfaces it has on each side. Due to this reason, the chunk located at coordinate 23 (row 2 and column

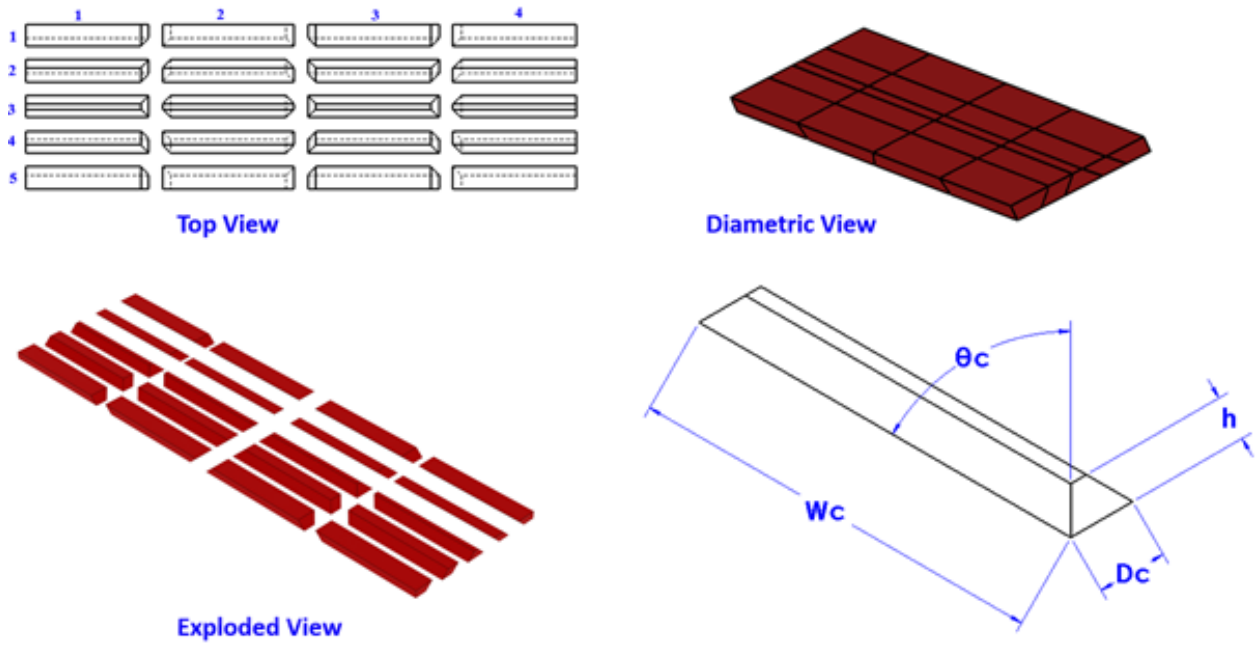

Figure 7.1: (a) Top exploded view of a part showing individual chunks (both rows and columns are numbered). (b) Dimetric view of the part showing chunk's boundary. (c) Dimetric exploded view. (d) Dimension of a center chunk).

3 in Figure 7.1(a)) has a different shape (positive slope on both horizontal ends whereas a negative slope on inside and positive on outside in vertical ends) than the central chunk. The exploded top view in Figure 7.1(a) shows the shape of each chunk at a different row and column location, whereas the dimetric view in Figure 7.1(b) shows the boundary lines on a part at which the division takes place. The dimension associated with the chunks are depicted in Figure 7.1(d). To ensure the printability of a chunk, the following constraints need to be satisfied.

1. As shown in Figure 7.2(a), if θ_c is the angle of the sloped bonding interface between the chunks, θ_e is the angle of the exterior of the extruder nozzle from the vertical, and h is the tallest height of the chunk, and the overall depth of each chunk is D_c, then θ_c

is guided by the following equations (7.1) and (7.2).

$$\theta_c \leq 90 - \theta_e \tag{7.1}$$

$$\theta_c \geq \tan^{-1} \frac{h}{0.5D_c} \tag{7.2}$$

Equation (7.1) must be satisfied. Otherwise, the nozzle, in Figure 7.2(a), will interfere with the printed part of the chunk; however, the lower level of the angle is limited by the height and the overall depth of the chunk. Printing angles lower than one specified by equation (7.2) would require either lowering the height of the chunk or increasing the depth of the chunk (which could be a physical constraint as the reach of the robot might be exceeded). Thus, the minimum value for slope angle that can be used for sloped surface chunking strategy is given by equation (7.2), which is bounded by the maximum value of D_c.

2. If the reach of the printhead arm is D_e, which is the lateral distance between the point of material extrusion (print nozzle) and the nearest part of the wheels and/or chassis of the robot (for mobile robots) or length of printing arm (for printers with arm such as SCARA, robotic arm, etc.), and the overall depth of each chunk is D_c, the following equation must hold true.

$$D_c \leq D_e \tag{7.3}$$

3. If the width of a mobile printing robot is W_r and the width of the chunk is W_c as illustrated in Figure 7.2(b), the following should be true in order to avoid potential

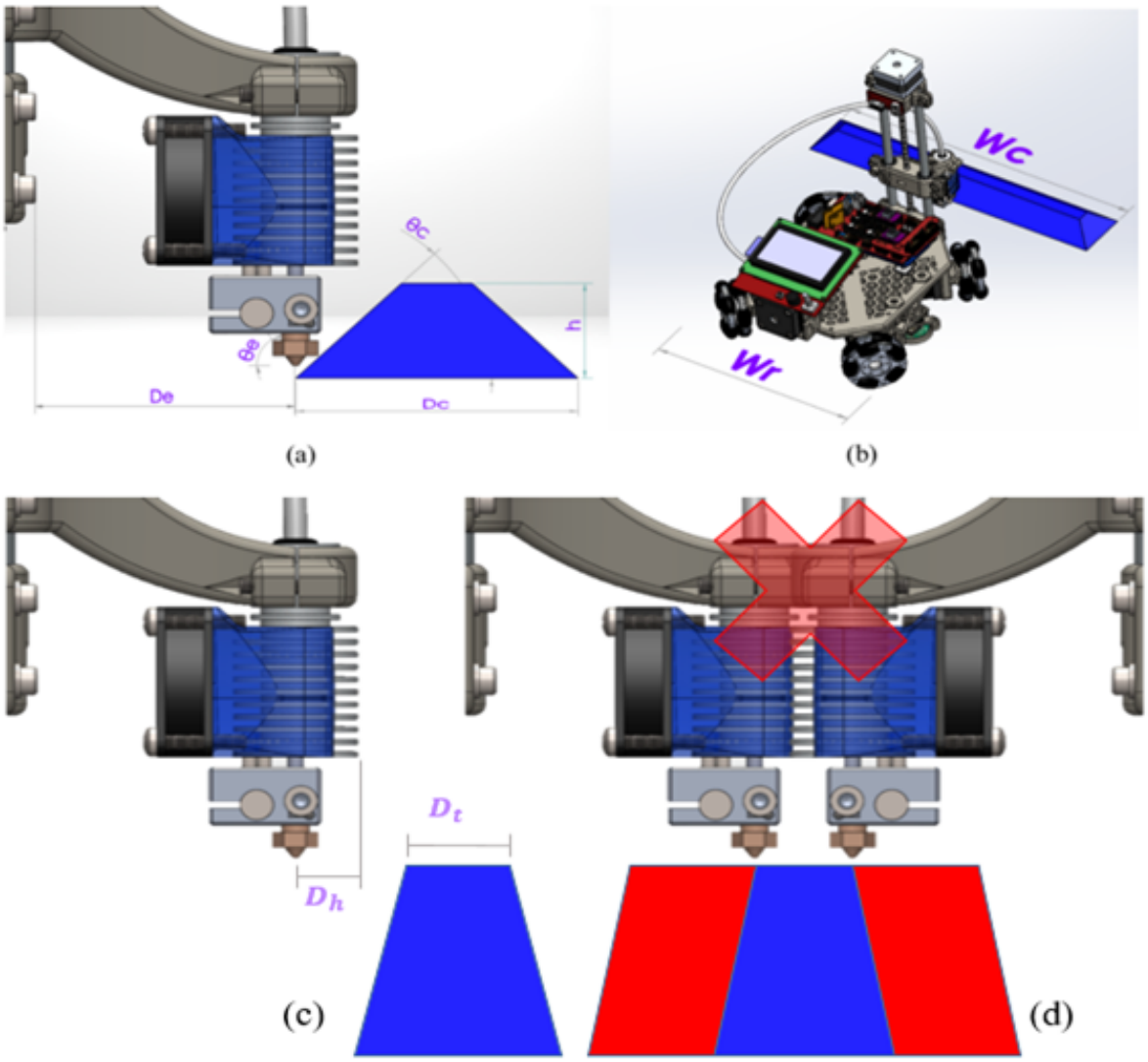

Figure 7.2: (a) Illustration of chunk's dimension and printing limitations on the slope, (b) Comparison of chunk width with the width of the robot, (c) Dimension of the top base of center chunk and distance between the nozzle and the hardware end and, (d) scenario showing a collision between the hardware when the top base of the chunk is too narrow to fit the hardware while printers are working on opposite rows of center chunk row.

collision between adjacent active robots in a row. This chunking constraint is only applicable to the chunking strategy if used in conjunction with SPAR3.

$$W_c \geq W_r \tag{7.4}$$

The sloped-surface chunking strategy divides the object into columns and rows. The number of columns n_{cols} directly determines how many robots can be used on each side of the center row for printing (i.e., 2 columns for 1 robot on each side of the center row). Based on the chunking constraints presented above and the number of robots available for printing, the total number of chunks can be determined with the following procedure.

1. Number of chunk columns (n_{cols}): The maximum number of chunk columns can be determined by dividing the entire length of the part by the smallest chunk width, which is equal to the width of the robot. Thus, if the length of a part is L and the width of the robot is W_r, then,

$$Max.\ number\ of\ chunks\ column\ (n_{cols}^{max}) = \frac{L}{W_r} \tag{7.5}$$

   Once the upper bound is calculated, the ideal number of chunk columns for a given number of robots (N) can be calculated using the equation below:

$$Numberofchunkscolumn(n_cols) = min(N, n_{cols}^{max}) \tag{7.6}$$

2. Number of chunk rows (n_{rows}): The number of chunks rows, on the other hand, is guided by the constraints related to the chunk depth (equation 7.3). Using the depth constraint, we calculate the minimum number of chunk rows for the part by dividing the width of the part by the largest chunk depth permitted, which is equal to the reach of the printhead arm of the robot. Thus, if the width of the part is W and the reach

of the printhead arm is D_e, then,

$$Min.\ number\ of\ chunk\ rows\ (n_{rows}^{min}) = \frac{W}{D_e} \tag{7.7}$$

The ideal number of chunk rows is not dependent on the total number of robots available for printing. For the SPAR3 strategy, a smaller number of chunk row is desirable in order to avoid unnecessary travel between the print sequences. Although, if the number of chunk rows is less than three, only half of the printing robots can be utilized for printing which diminishes the printing efficiency. On the other hand, it also needs to be ensured that the top base of the center chunk (D_t) is twice as wide as the distance between the nozzle and the end of the hardware (D_h). This is to avoid collision between the different printheads of the robots working on either side of the center chunk, as shown in Figure 7.2(d).

Once the number of chunk columns and number of chunk rows are calculated, the total number of chunks can be calculated using the following equation,

$$Total\ number\ of\ chunks = n_{cols} \times n_{rows} \tag{7.8}$$

As mentioned at the beginning of the chapter, the chosen chunking strategy should result in a part that does not compromise the overall integrity of the part. Tensile strength is one of the indicators of how strong a part is mechanically and a good measure of the part's integrity. Thus, we need to test the mechanical strength of the chunked parts using the sloped-surface chunking strategy, which is done in the subsequent section.

7.3 Introduction: Mechanical Strength of Parts Printed Using Sloped-Surface Chunking Strategy

In the existing literature, many studies are reported on the strength of the FDM 3D printed parts and their anisotropic behavior using the traditional layer-based 3D printing. For example, Ahn et al. performed a seminal study on the relationship between the strength and anisotropic behavior of the FDM part and various build parameters, such as air gap, raster orientation, bead width and, model temperature [99]. Rezayat et al. investigated the contribution of the infill materials to bear the load of the part and concluded that the infill materials do not contribute as much to the strength of the part as the perimeter materials based on a multiscale study [100]. Similarly, J. Chacon et al. conducted a study to characterize the effect of feed rate and layer thickness along with build orientation on mechanical strength [101]. They concluded that ductility of a part decreases as layer thickness and feed rate increases whereas, the mechanical properties get better as layer thickness increases but decreases with increase in feed rate. H. Kim et al. studied the impact of the bond strength at the interface between two different materials (ABS and PLA in this case) on the overall strength of a part printed with dual materials [102]. In their study, it was observed that there were voids and overlaps between the two-materials printed in vertical layers. It was concluded that such defects may affect the adhesion between two materials and weaken the part. They also concluded that adding vertical layers to a part might not be effective because of the defect mentioned earlier whereas adding horizontal layers improved the mechanical properties of the FDM part.

D. Espalin et al. conducted similar studies using multi-material extrusion using two

different FDM printers [103]. In their study, raster beads were printed using one printer with one type of material and the contours were printed using second FDM printer with different material. Along with this, they also ran some experiments with different material per layer, similar to the studies done by Kim et al. In the study, they concluded that the multi-material FDM part exhibits similar tensile properties as standard FDM part. They did, however, find that there is improvement in surface roughness of finished part using coarser infill but finer contours. They also concluded in the study that using two FDM printers instead of one, reduces the total print time by more than half. The multi-material FDM studies discussed above may seem similar to the chunk-based printing at first glance but are fundamentally different. Multi-material FDM is accomplished using traditional layer-based printing unlike chunk-based printing. Additionally, the tensile loading is in perpendicular direction to the multi-material printing whereas in chunk-based printing, it is in the same direction as the adhesion between the chunks. While these studies are helpful in understanding how the build parameters in the traditional layer-based 3D printing influence the mechanical strength of the printed part, little is known on how the chunk-based build parameters would impact the bond strength between chunks and the mechanical strength of the chunk-based 3D printed parts.

In this section, we will attempt to fill this knowledge gap by comparing the strength of the chunk-based printed parts against the standard FDM parts. We, first, identify the chunk-based printing parameters that can directly influence the bond strength between chunks, such as the chunk slope angle, the chunk overlapping depth, and the number of perimeter shells. We then conduct design of experiment to understand how various combinations of those identified parameters would affect the tensile strength of the printed parts. It however needs

to be clarified that it is out of the scope of my expertise to analyze the change in underlying material properties such as local properties at different location due to the changes of the printing parameters.

The overall strength of chunk-based 3D printed part will be influenced by both the traditional layer-based printing parameters and the additional chunk-based printing parameters. Based on whether or not the parameters have direct impact on the bond strength between chunks, they can be categorized in two groups: a) Direct parameters, and b) Indirect parameters. In this section, a summary of both direct and indirect parameters that are considered and tested in this study is provided.

7.3.1 Direct Parameters

In this preliminary study, three parameters were identified to be directly influential to the strength of the chunk-based printed parts.

1. **Chunk Slope Angle**: Considering the characteristics of the FDM process, the chunks are not divided by a vertical plane, but a sloped plane, as illustrated in Figure 7.3, such that the printhead can deposit materials on the chunking plane to facilitate the bonding between chunks. Therefore, the slope angle of the chunking plane will have direct impact on the bonding strength between chunks.

2. **Number of Perimeter Shells**: As indicated by Rezayat et al. in their study, the perimeter shells carry much more load and contributes much more to the strength of the printed part than the infill materials in the traditional layer based FDM printing [100]. Therefore, the presence or absence of a perimeter shell at the chunking plane could

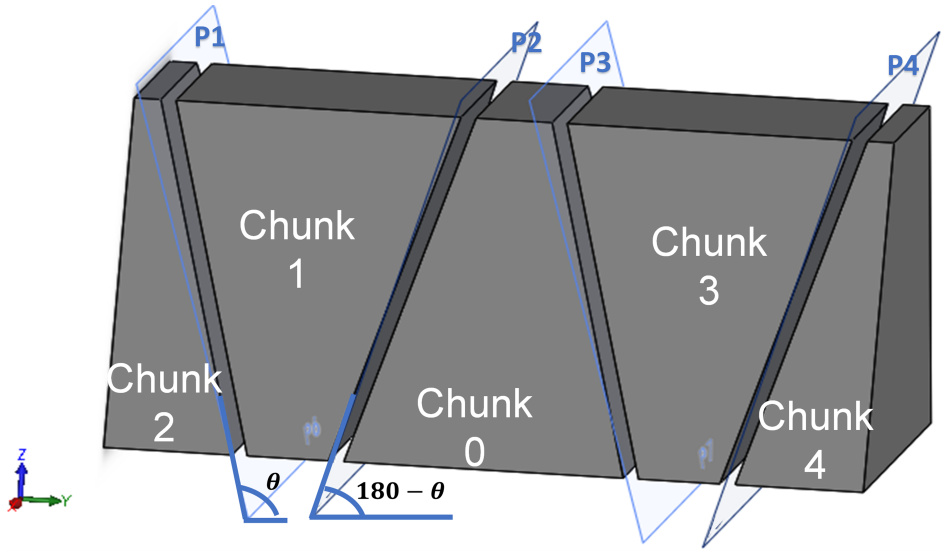

Figure 7.3: Illustration of the chunking plane between chunks.

significantly impact the bond strength between the chunks. As illustrated in Figure 7.4, the presence of a perimeter (Figure 7.4(b)) results in uniform contact between the chunks, whereas the absence of it (Figure 7.4(a)) creates voids and overlaps in the interface, thus adding additional uncertainties to the strength of chunk-based printed parts.

3. **Chunk Overlapping**: In the same way that FDM overlaps the filament to increase the infill density of the print, the chunks can also be slightly overlapped to improve bonding strength as illustrated in Figure 7.4(c) and 7.4(d). If the chunks are printed exactly along the chunking plane, the overlapping is zero. A positive overlapping means more materials are squeezed into the chunking plane and will make the contact area between chunks denser. A negative overlapping indicates the chunks are not in contact with one another.

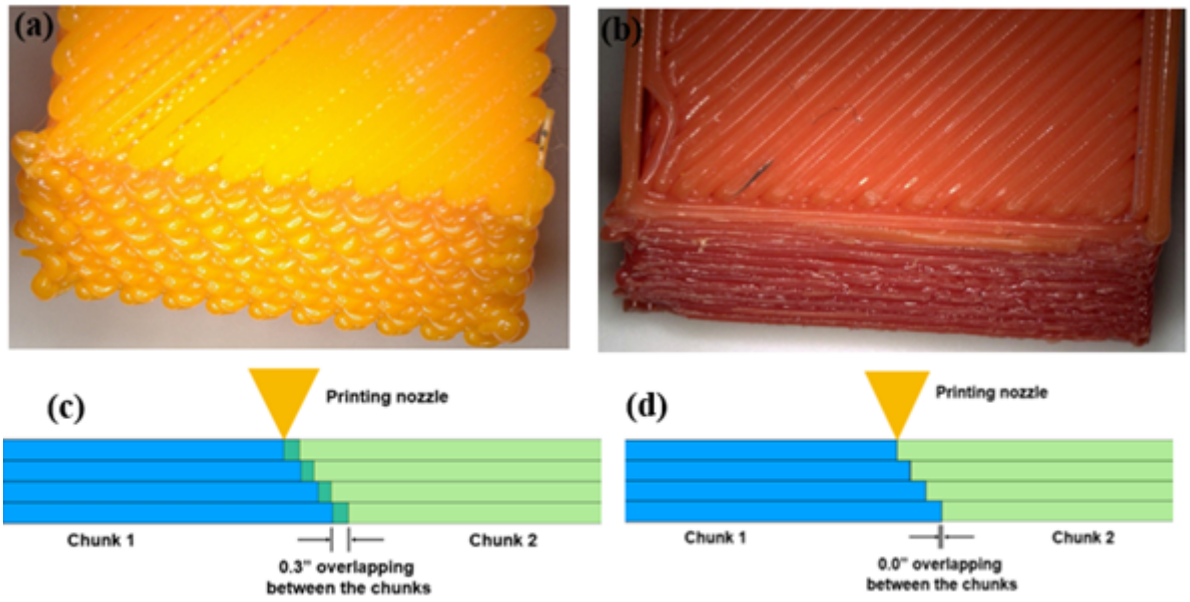

Figure 7.4: (a) Chunk printed without perimeter shells (b) Chunk printed with perimeter shells (c) Chunks with 0.3mm overlapping between their layers (d) Chunks with 0.0mm overlapping between the layers.

7.3.2 Indirect Parameters

Other than the identified parameters that are directly related to the chunk bond, the traditional layer-based printing parameters, such as raster orientation, infill density, air gap, bead width, print temperature, etc., also influence the overall strength of the chunk-based 3D printed parts. The most important indirect parameters are listed below.

1. Raster orientation: Raster orientation is the angle orientation of road or bead compared to the tensile loading. The default setting of 45°/-45° was used for this experiment. Raster orientation of 90° has the lowest tensile strength and 0° has the highest one [99].

2. Infill density: Infill rate is the percentage of material inside the part. Higher the infill density means that more material is contained inside the part. This makes a part stronger and sturdier. On the other hand, lower infill density results in parts that are

more fragile, but printing time is shorter. 100% infill rate is used as default for all the samples in this study.

3. Air Gap: Air gap is the distance between two beads in traditional layer-based printing. Increasing the air gap weakens the part because it creates spaces between the beads. On the other hand, using negative air gap, enhances the strength of the parts as it makes the structure denser. The default value used in this study is zero.

Indirect parameters and their impact on the tensile strength of traditional layer-based parts has been thoroughly studied in existing literature, and it is assumed that those parameters will have a similar impact on the tensile strength of chunk-based 3D printed parts. Therefore, the main focus of this paper will be on the effects of the direct parameters on the overall strength of the chunk-based parts and the indirect parameters are kept constant. Prior to the experiment, the following hypothesis were postulated about the relationship between each of the direct parameters and the tensile strength:

1. The lower the slope angle, higher is the tensile strength between the chunks. Lower slope angle means larger contact area for the same thickness of chunks. This could result in higher tensile strength. Thus, it is predicted that the chunks with 30° slope will have stronger tensile strength than that with 50° slope.

2. Absence of perimeter shells in bond area makes a chunk-based part stronger. Presence of shells imitates behavior similar to 90° raster orientations between the infill raster and the boundary. Existing study [99] has shown that 90° raster orientations lead to weaker tensile strength. So, it is predicted to have stronger bond between the chunks that are printed without perimeter shells.

3. Higher the overlapping between the layers of the adjacent chunks, higher is the tensile strength. More overlapping implies more materials from two chunks are fused together. Therefore, it is expected that lager overlapping will produce a stronger bond between chunks as compared to smaller overlapping.

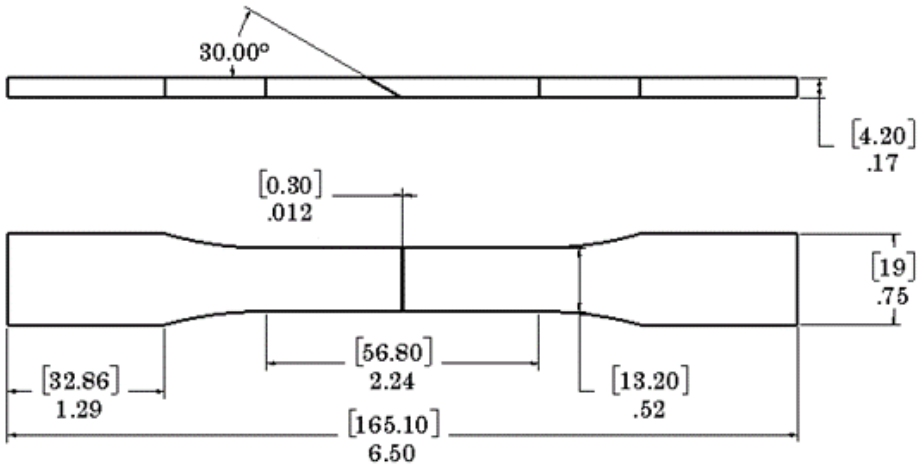

Figure 7.5: Sample geometry with chunk overlapping of 0.3mm and 30° slope angle.

7.4 Experimental Setup and Methodology

To assess the impact of the direct parameters on the bond strength, three steps are required. First, the specimen needs to be fabricated. Second, a systematic design of the experiment is necessary. Third, the tensile tests need to be conducted on the fabricated specimens. In this section, we describe the three steps in the sequence of their execution.

7.4.1 Specimen Fabrication by Chunk-Based Printing

Although chunk-based 3D printing is originally developed for cooperative 3D printing with mobile 3D printers, it can be applied to existing FDM 3D printers because the depositing

Table 7.1: Printing parameters for fabrication of the samples

Printing Parameters	Values
Filament material	PLA
Printhead temperature	215 ° C
Print bed temperature	80° C
Nozzle diameter	0.4 mm
First layer thickness	0.26 mm
Subsequent layer thickness	0.2 mm
Printing speed	28 mm/s
Infill printing speed	42 mm/s
Raster angle	45/-45°
Infill density	100%
Air gap	0 mm

process for both techniques are the same. At the time of this experiment, the mobile 3D printers were still under development and their reliability and consistency was not sufficient, for this study, a commercially available Ultimaker 3D printer was used to print the specimen chunk by chunk. To achieve this, the printing path needed to be altered so that the resulting paths mimic the printing process of two robots. In this study, the slicing software Simplify3D, was used to generate the printing paths, slicing the chunks and generate the corresponding G-code. A test coupon (Figure 7.5) was first designed according to ASTM D638 Type I

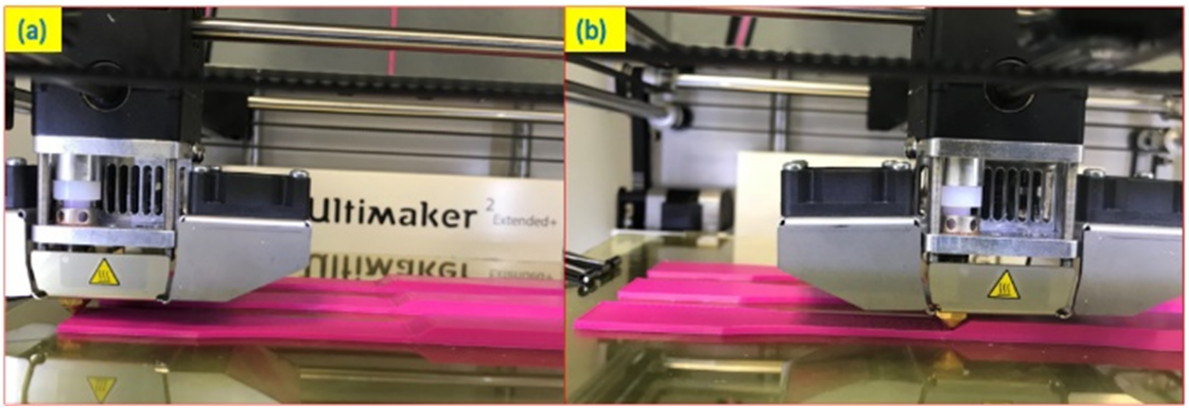

Figure 7.6: Chunk-based printing using Ultimaker 3D. The printer completes the right chunk first and then starts printing the left chunk.

Standard using SolidWorks. The test coupon was then divided into two equal chunks with

Table 7.2: Range of Direct Parameters

Variable	Abbr.	Low Level	High Level
Chunk Slope Angle (°)	SA	30	50
Chunk Overlapping (mm)	CO	0.3	0.5
Number of Shells (#)	NS	0	2

specified chunking parameters (e.g., slope angle, location of the chunking plane, etc.). The G-code files were then manipulated such that Ultimaker prints the left chunk first with a sloped chunking plane at the end, and then prints the right chunk as shown in Figure 7.6. The printing parameters are listed in Table 7.1.

7.4.2 Design of Experiment

To study the impact of the direct parameters on the strength of the chunk-based 3D printed parts, we need to properly define the range of parameters for the samples to be fabricated and tested. Preliminary tests were conducted to investigate the correlation between the direct parameters and tensile strength. These tests were done by altering one variable at a time while keeping the rest of them constant.

The two values used for slope angles were 30° and 50°. At the time of testing, we could not find nozzles on the market that can print on a slope angle larger than 50°, thereby this angle is set as the upper bound. Any slope angle below 30° limits the height of the chunk, so they are ignored [58]. Chunk overlapping of 0.3mm and 0.5mm are selected. The reason for choosing 0.3mm is that this value is equal to the width of a single "raster". To understand whether the presence of shells would affect the strength, 0 and 2 shells were chosen, 2 being the default setting of perimeter shells and 0 being absence of shell altogether. Table 7.2 summarizes all the design variables and the ones chosen for the experiment. To maximize the test coverage, orthogonal arrays were created. Three parameters with two

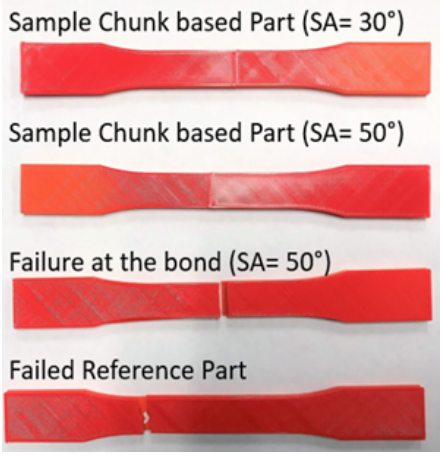

Figure 7.7: Test specimens.

levels of values provided eight different specimens. The orthogonal arrays of 8 specimen variations are provided in Table 7.3. Columns 2 – 4 represent the test levels of the factor. Each of the rows represents the test runs. The table was used as the plan for multi-factorial experiments to detect the effects of various direct parameters on the tensile strength. Five samples were printed for each combination and five tests were performed, respectively, as per ASTM testing standard.

Table 7.3: Design of experiments for the three direct parameters

Specimen Index	**Chunk Angle (SA)**	**Chunk Overlapping (CO)**	**No. of Shells (NS)**
1	Low	Low	High
2	Low	High	Low
3	Low	High	High
4	Low	Low	Low
5	High	High	High
6	High	High	Low
7	High	Low	High
8	High	Low	Low

7.4.3 Testing Setup

The tensile test was conducted on MTS® with MTS microcontroller®. A 5 kN loadcell was used to load the samples in tension. The samples were loaded at the rate of $2mm/min$, and each sample was loaded until the failure occurs. Ultimate tensile strength was chosen as the measure of mechanical properties. In the test, all the failures took place at the chunk joint, so we were assured that the test results measured the strength of the chunk joint. The macroscopic view of the failure along with samples is shown in Figure 7.7.

7.5 Results and Discussion

7.5.1 The Effect of Chunk Angle

The result of the tensile test for three different slope angles (30°, 45°, and 50°) is plotted in Figure 7.8. Since two points (30° and 50°) are not sufficient to express the functional relationship between the angle and the tensile strength, we added another data point (45°) to the graph. Five iterations of printing and testing were done for each slope angle. The other two parameters are controlled, and all the samples have two perimeters shells and 0.3mm chunk overlapping depth. For the indirect parameter settings, the samples have 100% infill, 45°/-45° raster angle with no air gap between the beads. The results indicate that the chunk bond with slope angle of 30° fails at the average tensile strength of 32.42 MPa and the strength decreases as the slope angle increases. The strength of the samples with 30° chunk slope is about twice as much stronger as the one with 50°. This trend could be explained by the possible increase or decrease in the bonding area due to the change in slope angle. As a result of the smaller chunk angle, the length of the slope increases, which in turn leads to

larger overall bonding area. Larger bonding area results in an increase in number of bonds formed between two chunks making the part stronger. Similarly, an increase in chunk angle results in a reduction of length of slope and thus the smaller bonding area. This leads to a decrease in the number of bonds between the chunks, making the part weaker. To validate

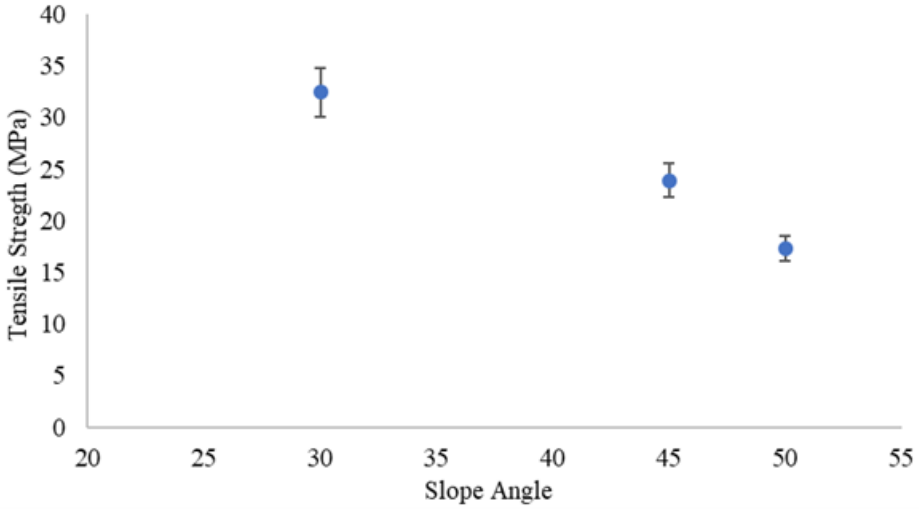

Figure 7.8: Tensile strength of specimen with different chunk slope Angle.

Table 7.4: t-Test result for slope angles

t-test assuming unequal variances (Welch's t-test)	**50° SA**	**30° SA**
Mean	24.81	32.57
Variance	37.72	3.24
Observations	20	15
One-tail P-value	9.75e-06	
t Critical one-tail	1.71	

the hypothesis proposed in Section 7.3, a Welch's two-sample t-test was performed on the chunks with 30° slope angle and the ones with 50°, as shown in Table 7.4. The one-tailed P-value (< 0.05) indicated the strength of the parts printed with 30° chunk slope angle is statistically significantly higher than that with 50° chunk angle.

7.5.2 The Effect of the Number of Perimeter Shells

Upon closer inspection, the failure of parts with shells occurred between the infill and the shells and not at the bond between the two chunks (Figure 7.9) essentially. This is because the shells are oriented at 90° with the loading direction, which decreases the strength in that region. One way to improve the strength is to increase the outline overlapping between the shell and infill so that a better contact between the infill and shell can be achieved. However, the default value of zero was used for outline overlapping for all experiments. The

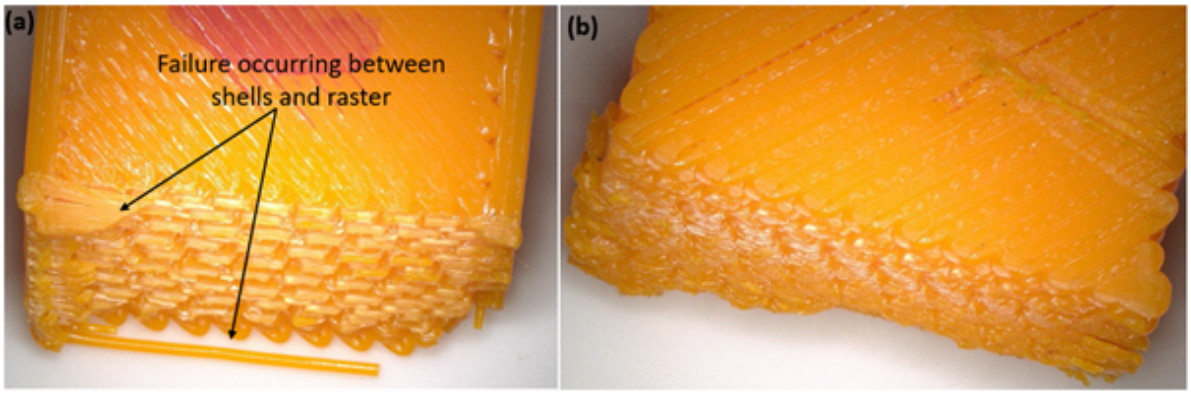

Figure 7.9: The failure occurred at the chunk joint. (a) Chunk printed with two shells around contact area (b) Chunk printed without shells around contact area.

general data trend in Figure 7.11 suggests that the chunk-based parts without perimeter shells have tensile strength higher than the ones with two perimeter shells except for sample 4 (SA=30°, CO=0.3, NS=0). This anomaly is due to the fact that the bond between the chunks is not properly formed at the bottom of the specimen, as shown in Figure 7.10. Even though the overlapping is set to 0.3mm, the chunks are not well bonded at the bottom as compared to the top surface. This reduction of effective contact area between chunks results in significantly lower tensile strength.

In order to better investigate the influence of shells (absence or presence) on the

Table 7.5: t-Test result for number of perimeter shells

t-test assuming unequal variances (Welch's t-test)	**No Shells**	**Two Shells**
Mean	31	25.99
Variance	11.88	47.23
Observations	15	20
One-tail P-value	0.004	
t Critical one-tail	1.70	

tensile strength, a Welch's t-test was conducted as well. Due to anomalous behavior of sample 4, it was treated as an outlier and was excluded from the hypothesis testing. The analysis of the t-test is provided in Table 7.5. The result shows that there is a statistically significant difference between the chunk-based part printed with shells and the ones printed without shells.

7.5.3 The Effect of Chunk Overlapping

It is intuitive that the bond between chunks become stronger as chunk overlapping increases. Using the graph presented in Figure 7.11, we compared the strength of the chunk bonds based on the chunk overlapping keeping the rest of the parameters constant. The general data trend in Figure 7.11 shows that with the increase in overlapping, the strength of the chunk bond increases as well except in the case of sample 1 and sample 3. Both samples 1 and 3 have other parameters constant ($SA = 30, NS = 2$) the only change being the chunk overlapping. Sample 3 has 0.50 mm overlapping whereas sample 1 has 0.30 mm overlapping but the sample 1 shows higher tensile strength than that of sample 3. Even though the difference is smaller than 1 MPa (0.6 MPa), it is contrary to the hypothesis we made. On the other hand, the rest of the samples (4 and 2, 7 and 5, 8 and 6) supports the made hypothesis. This leads me to believe that samples 1 and 3 represent anomaly

Table 7.6: t-Test result for chunk overlapping

t-test assuming unequal variances (Welch's t-test)	**0.3mm**	**0.5mm**
Mean	22.75	31.22
Variance	41.22	8.41
Observations	20	20
One-tail P-value	6.23604e-06	
t Critical one-tail	1.71	

rather than the trend. In order to confirm this, we ran a Welch's two-sample t-test similar

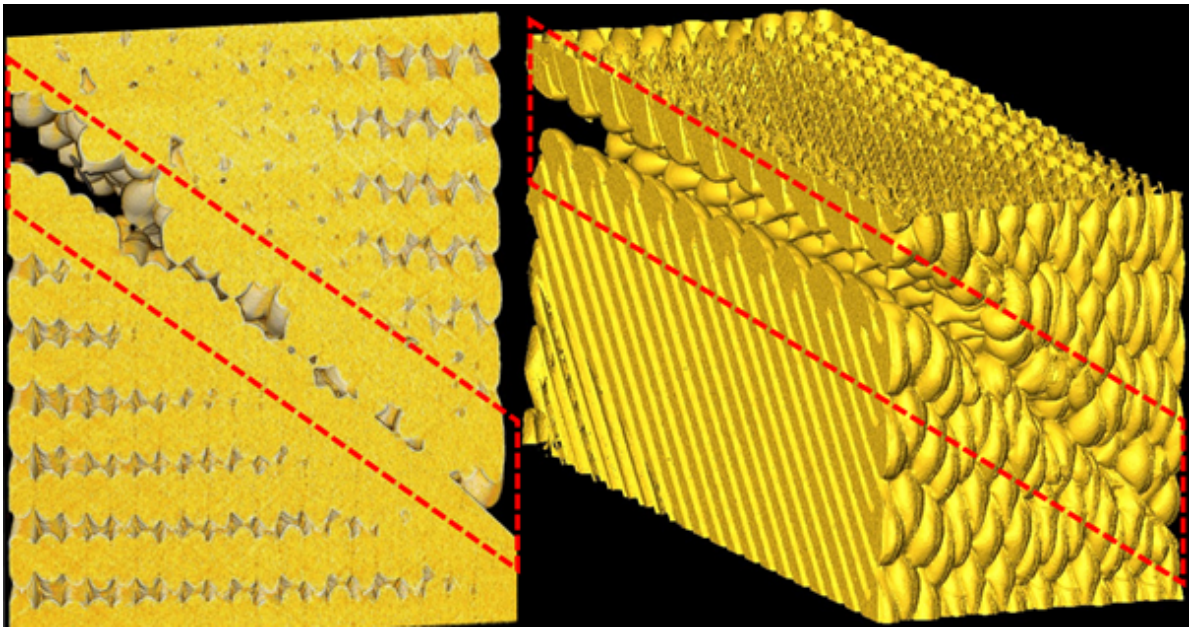

Figure 7.10: The gap between two chunks on sample 4 at the bottom layer. The gap at the top layer is significantly smaller. This gap results in reduction of the effective contact between the chunks resulting in weaker chunk bond.

to the one we ran for other two parameters (i.e., the slope angle and perimeter shells). The results are presented in the Table 7.6 and show that the strength of the parts with 0.30 mm overlapping is statistically significant different ($p - value < 0.05$) from the strength of the ones with 0.50 mm overlapping.

7.5.4 Surface Failure

In the tests, all samples without perimeter shells failed at the chunking surface (samples 2, 4, 6, and 8). On the other hand, as mentioned earlier, the presence of shells resulted in failure at the transition between the shell and infill upon closer inspection. It can be observed that the failure in first case is due to the breaking of the filament. As shown in Figure 7.9(b), a crack started at the weakest point of the bond but then propagated vertically upward, breaking the filament rather than following the adhesion boundary between the chunks. On the other hand, the failure in the latter case is due to the de-bonding between the raster and shell, as shown in Figure 7.9(a).

7.5.5 Chunk-based Printing vs. Layer-based Printing

Figure 7.11 shows the results of the tensile test for the orthogonal arrays presented in Table 7.3. Each of the specimens have the same indirect parameters (100% infill, 45°/-45° orientation, no air gap). The reference or control sample part is printed as a single piece. The tensile strength of the reference part was 28.23 MPa, as shown in Figure 7.11. Specimen 1 ($SA = 30, CO = 0.3mm, andNS = 2$), specimen 2 ($SA = 30, CO = 0.5mm, NS = 0$), specimen 3 ($SA = 30, CO = 0.5mm, NS = 2$), and specimen 6 ($SA = 50, CO = 0.5mm, NS = 0$) failed at tensile strength higher than that of the reference part. Specimen 2 had the highest tensile strength on average among all the specimens (34.05 MPa). Therefore, this leads to my preliminary conclusion that with appropriate combinations of the chunk-based printing parameters, we might be able to obtain a part with a strength that is as high as that of traditional 3D printed parts, if not higher. On the other hand, specimen 4 ($SA = 30, CO = 0.3mm, NS = 0$), specimen 5 ($SA = 50, CO = 0.5mm, NS = 2$), specimen 7

$(SA = 50, CO = 0.3mm, NS = 2)$, and specimen 8 $(SA = 50, CO = 0.3mm, NS = 0)$ failed at 18.95 MPa, 26.76 MPa, 17.34 MPa, and 21.84 MPa respectively. Those values correspond to 67, 95, 61, and 77 percent of the reference part's tensile strength respectively. Specimen 7 had the lowest tensile strength of all the test specimens (17.34 MPa). Therefore, for chunk-based 3D printing, it is recommended to avoid setting high chunking angle, low overlapping depth, and large number of shells at the same time, as this combination could result in chunk-based part that is significantly weaker than the traditional 3D printed part. Based

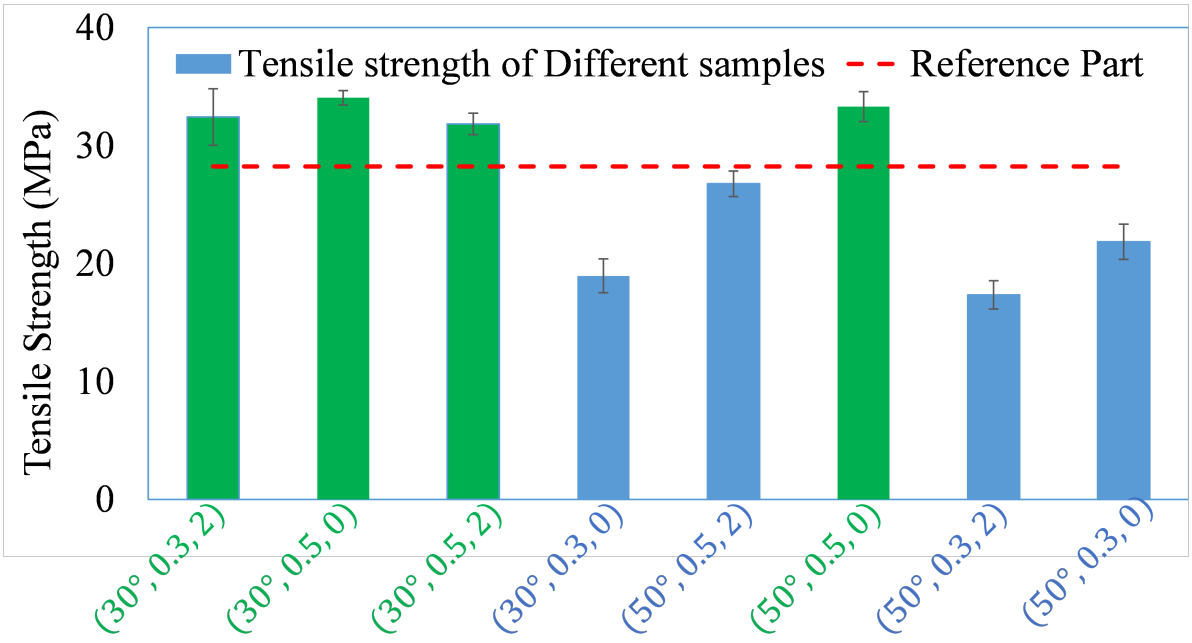

Figure 7.11: Tensile strength (MPa) of the specimens printed with various combination of chunk-based parameters in Table 7.2 along with tensile strength of reference part.

on this experimental study, it can be concluded that proper selection of the combinations of the chunk-based printing parameters makes chunk-based parts as strong if not stronger than traditional layer-based parts, while some combinations could make it weaker. The following recommendations regarding chunk-based parameters are provided:

- **Use smaller slope angle to strengthen a part**. Tensile strength decreases with the increase in chunking slope angle. This is due to increased contact area between the chunks.

- **Avoid shells at the contact area between chunks to increase strength**. Printing chunks without any perimeter shells at the bonding surface strengthens the adhesion between chunks. This is due to the fact that presence of shells mimics the 90° raster orientation to the tensile load, which have the lowest tensile strength. If the shells are to be printed at the bonding area, the outline overlap can be increased to somewhat improve the strength.

- **Increase chunk overlapping to strengthen a part**. Overlapping increases the strength of the part. It is important to bear in mind that increasing the overlapping will affect the overall dimension of the part in overlapping direction and there is only so much overlapping that can be done before printing becomes infeasible.

7.6 Conclusion

In this chapter, tensile specimens using PLA were fabricated with varying chunk-based parameters to investigate how those parameters would affect the tensile strength of chunk-based 3D printed parts. A comparative study was performed to understand how the strength of those chunk-based printed parts collated with the standard FDM parts, i.e., the ones printed normally layer by layer. Design of the experiment was conducted to understand how parameters such as the chunk slope angle, overlapping depth, and the number of perimeter shells affect the tensile strength of chunk-based printed parts. The tensile strength of chunk-

based parts ranged from 121 to 61 percent of the tensile strength of the standard FDM part. Based on our study, it is found that proper selection of the combinations of the chunk-based printing parameters makes chunk-based parts stronger than traditional layer-based parts, while some combinations could make it weaker. The following recommendations regarding chunk-based parameters are provided:

- Use smaller slope angle to strengthen a part. Tensile strength decreases with the increase in chunking slope angle. This is due to increased contact area between the chunks.

- Avoid shells at the contact area between chunks to increase strength. Printing chunks without any perimeter shells at the bonding surface strengthens the adhesion between chunks. This is due to the fact that presence of shells mimics the 90° raster orientation to the tensile load, which have the lowest tensile strength. If the shells are to be printed at the bonding area, the outline overlap can be increased to somewhat improve the strength.

- Increase chunk overlapping to strengthen a part. Overlapping increases the strength of the part. It is important to bear in mind that increasing the overlapping will affect the overall dimension of the part in overlapping direction and there is only so much overlapping that can be done before printing becomes infeasible.

While the study in this chapter demonstrates the viability of chunk-based printing for C3DP, it is essential to understand how these identified parameters change the material properties, which in turn impact the mechanical properties of the chunk-based parts. In addition to this, how do these chunking parameters that impact the mechanical properties

of the chunk-based parts influence the schedule planning in C3DP? While the impact of the direct parameters is studied in this chapter, the impact of the indirect parameters on the material properties of the chunk-based part is yet to be studied. Even though several studies have been done on the effects of these indirect parameters in the conventionally 3D printed part that could provide accurate guidance, there might be a new revelation that is yet to be explored. Hence, the amalgamation of the direct and indirect parameters could impact the planning of scheduling of multi-robot printing in a manner that is yet unknown. For example, a chunk printed with a particular combination of direct parameters (slope angle) with specific indirect parameters (e.g., infill orientation) could yield a better part with good mechanical properties if specific chunks are printed before the other chunks. Similarly, a different print sequence for the same combination of direct and indirect parameters could yield a part with poor mechanical properties. A detailed study is required to understand such correlation between the mechanical properties and planning in C3DP and could be part of future work.

8 CONCLUSION AND FUTURE WORK

8.1 Contributions

While many manufacturing paradigms have been proposed over the last decade, as the hardware and software capabilities expand in manufacturing, they still focus on traditional manufacturing factories, where only a single type or a single family of product is manufactured. On the other hand, Swarm manufacturing focuses on building factory floors within a local ecosystem where multiple products can be produced to meet the changing customer demand from mass manufacturing to the production of customized products. Rather than focusing on improving the efficiency of mass production, swarm manufacturing focuses on new customer demand for mass customization. Cooperative 3D printing (C3DP), one of the enabling technologies to achieve swarm manufacturing, is a novel approach to additive manufacturing, where multiple mobile 3D printing robots work together cooperatively to print the desired part, representing a major step towards swarm manufacturing. At the core of C3DP lies the chunk-based printing strategy. This strategy splits the desired part into small chunks. The chunks are then assigned and scheduled to be printed by individual printing robots.

This project demonstrates the first application of multiple mobile robots for additive manufacturing applications to the best of my knowledge. In doing so, the work has demonstrated how to discretize the process of additive manufacturing that will allow the use of multiple robots for printing, and simultaneously eliminate the need for post-processing assembly operations such as gluing of the chunks. This is because the bond between the

chunks is formed during the printing process as the new material is deposited on the sloped surface of already printed chunks. The mechanical study of the bond formed in chunk-based printing demonstrates that a part printed using chunk-based printing can result in part as strong as the printed using conventional 3D printing.

The result of discretization of the process of additive manufacturing results in multiple stages that are interdependent including chunking, scheduling, and path planning. In this thesis, I primarily focused on scheduling strategies for multi-robot C3DP. The work in the scheduling of multi-robot C3DP ranges from the development of the first working scheduling strategy based on human heuristics to a decentralized approach for multi-robot planning that uses local information to search and print chunks. In addition to presenting frameworks for collision-free printing in C3DP, several key components were identified that could be vital in validating the new developments in C3DP scheduling. Evaluation metrics using a directed dependency tree (DDT) can be used to compute the total time it takes to print an object, whereas, geometric constraints checks using the swept volume and accessible space of the robots and the occupied space of the chunks, can be used to validate any print strategy developed in the future.

In addition to this, this thesis also presents a planning approach for C3DP printing, which incorporates scheduling and path planning using a decentralized approach. In the decentralized planning approach, printing robots act as autonomous agents and make decision based on local sensing. The developed decentralized approach is based on a simple set of rules that have to be followed by each individual printing robot. Such an approach is very resilient to uncertainties, which is prevalent in additive manufacturing. Such resiliency was demonstrated using an example of uncertainity in print time, highlighting the fact that

the estimated print time and actual printing time rarely match in AM. The decentralized approach based on a simple set of rules demonstrates no adverse impact as a result of such uncertainty. A centralized approach on the other hand, could result in failure since the estimated time and the actual print time do not match and the planning in such approach are done based on the estimated print times.

8.1.1 Revisiting Research Questions

At the beginning of this dissertation, we posed two research question. RQ1: How can the traditional 3D printing process be transformed to enable multi-robot cooperative AM? RQ2: How can cooperative manufacturing planning be realized in the presence of uncertainties in addition to spatiotemporal constraints that are dynamic in space and time? To answer RQ1, we discretized the entire process of C3DP into multiple interdependent stages. We then developed approaches to address each of the individual stages. To enable multi-robot cooperation, we partitioned a part into multiple chunks and studied the mechanical strength of the chunk-based part in Chapter 7. The study reveals that the mechanical strength of the chunk-based parts can be as high as that of part printed using layer-based printing. We then developed multi-robot cooperative scheduling strategies that allow simultaneous multiple robots without causing any collision with one another or with the already printed part. Three different scheduling approaches are presented. First, a simple strategy is used, based on heuristic to obtain a working strategy that would allow such collision-free cooperation. Second, an approach to automatically generate a print schedule for any given part was developed. The approach searches the larger design space and rejects print schedule that results in collisions, and only outputs valid print schedules that result in collision-free

C3DP. Finally, the third approach generates print schedules in scenario where resources are scarce. It uses meta-heuristic approach such as MGA and a novel constraint satisficing algorithm to minimize the total makespan, that can obtain near-optimal solutions for problems in C3DP. To answer RQ2, we developed a decentralized planning approach that is based on simple set of rules. Each printer involved in the printing acts as an autonomous agent and makes decision based on a simple set of rules. Using such approach, agents can make decisions based on the local information available to them. Such an approach to C3DP planning, as demonstrated by the results in Chapter 6, is very resilient to inherent uncertainties in AM (such as the difference between the estimated and actual print time). Thus, both the questions asked at the beginning of the research were successfully answered. While the said research questions were answered, new interesting avenues for future work have emerged from the research that can provide blueprints for how swarm manufacturing can move forward. Some of the possible future directions regarding this project are outlined in the subsequent section.

8.2 Future Works

In this section, I briefly discuss the possible future direction, including the use of learning algorithms, cooperation between the robots at the lower level.

8.2.1 Implementation of Learning-Based Approach for Multi-Robot Additive Manufacturing

While implementing a decentralized approach based on a simple set of rules has been implemented for C3DP, such an approach theoretically cannot guarantee an optimal solution.

Reinforcement Learning models can be useful for operating in a dynamic environment with many uncertainties and even obtaining optimal results. However, they suffer from extremely slow training times. However, with recent accelerated development in hardware such as GPU has provided new opportunities that can handle huge amounts of computation time necessary for training reinforcement learning algorithms (RL models). This can allow RL models to learn a control policy in a reasonably short period of time. Such models allow each agent to make rational decisions based on the information available to them in the face of uncertainties. However, it is important that the states and actions be carefully defined as the number of states and action determines the scale of the problem and determine the success of the approach.

A deep RL can be implemented using an actor-critic policy optimization model, where there are policy networks (actor-network) and value networks (critic network). Decentralized actor-network can return a policy distribution of actions and uses a deep neural network (DNN) to approximate the policy gradient and only has access to its own actions (action of individual agents). On the other hand, the centralized critic network returns an approximation of the value function and uses DNN to approximate the value function. The critic network has extra information (information of every agent's policy) that can be used to calculate the loss, which is to be minimized. Constraints and rewards can be organized such that it:

- minimizes travel time (add a negative reward for each movement)
- avoids collision before it happens (define the radius for each agent, and if the sum of the radius is less than a predefined critical value, it gets a negative reward)

- rewards printing of chunk (get baseline reward for printing any chunk, gets higher reward for printing chunks with a larger number of dependent chunks)

8.2.2 Implementation of Layer-level C3DP

Cooperation between the robots while printing a chunk could take place at the different granularity of time. At the highest level, cooperation between the robots takes place at the chunk level, followed by the layer level cooperation and at the lowest level, G-code level. In chunk-level cooperation, a part to be printed is discretized into chunks and the planning for multirobot cooperation is done by assigning the entire chunks to the robots for printing. The geometric constraints identified in chapter 3 are to be checked to ensure that no collision occurs between the robots while they are printing the entire assigned chunk. However, if we have the structure shown in Figure 8.1, the cooperation at the chunk level could not provide us with a valid print strategy because two tiles (individual components of the Delaunay loft are called tiles [104]) are intertwined with each other. Thus, no viable solution can be obtained if we simply rely on chunk-level cooperation. We need to break the tiles down into smaller components so that cooperation can take place at that scale. Thus, we discretize these two tiles into multiple layers, as shown in Figure 8.1 (b). The printing robots can cooperate with each other while printing the individual layer of the tiles and be able to print the whole loft with two tiles without colliding with each other. Such a situation is possible because, as outlined in Chapter 3, the violation of the geometric constraints check using swept volume means there is potential for collision, and the collision is not guaranteed (Figure 3.7(a) and (b)). Furthermore, if there is a violation of geometric constraints at the layer level, each layer can be further discretized into a G-code line segment, as shown in

Figure 8.1(c). And the cooperation between the robots can occur at the individual G-code line segment. For example, Figure 8.1(d) shows a single layer of two tiles of the Delaunay loft from Figure 8.1(a). Since both are regular hexagonal shapes, if we start printing tile 1 at position 1 and tile 2 at position 2, the printing robots will not collide with each other even though the geometric constraints are violated at the chunk level and layer level. This is because the constraints are not violated at the G-code level. This shows that cooperation between the robots at a lower level allows printing even if cooperation at a higher level is not possible; however, cooperation at a lower level requires closer control of the robot's movement while printing, printing trajectory. Such cooperation at a lower level could improve the overall mechanical strength of the parts printed using C3DP as the bond adhesion between the two structures take place at the layer level. Since the layers are stacked on top of one another, the layer above it will reinforce the bond strength.

Research in this direction could open up new avenues as space-filling tiles could be used to break a larger part into smaller geometrical structures such as the Delaunay Lofts. Such structures have self-interlocking features, which could further reinforce the mechanical strength of the printed parts. However, such research might require multi-level geometric partitioning where at a higher level, the entire space is divided into multiple regions, which can be assigned to individual robots for printing. At the lower level, the boundary region where robots collaborate can be further subdivided. While the rest of the space can be printed in a conventional manner (each region assigned to an individual robot), the boundary region has to be carefully navigated as that is the area where two robots might collide with each other. Thus, a novel research framework is required that could divide the border region to allow cooperation between the robots.

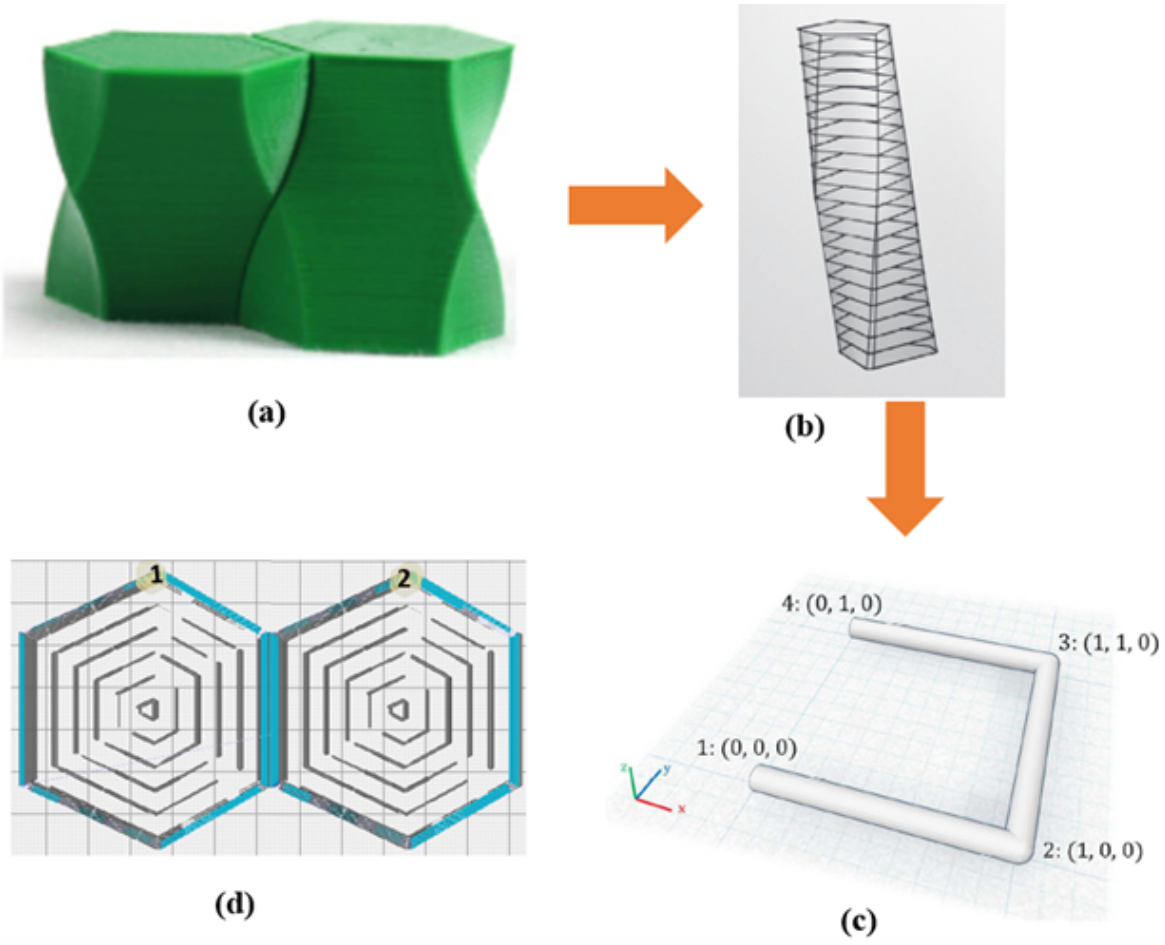

Figure 8.1: (a) Delaunay Loft tiles [104] (b) Decomposition of individual tiles into layers [104] (c) Decomposition of individual layer into G-code line segments [58] (d) Cooperation between the robots while printing line segment of individual layers of Delaunay lofts.

8.2.3 Design for Swarm Manufacturing

As mentioned in the previous section, given a part to be 3D, it can be chunked into multiple chunks using the chunking strategy and assigned to individual robots for printing. How can this approach be extended to swarm manufacturing? SM involves more than one manufacturing approach; how can we extend the framework to a multi-manufacturing approach? For example, assembly is a fundamental type of manufacturing operation that can significantly expand the capability of C3DP to SM. Therefore, integrating a digital assembly process in heterogeneous production is of great value. To integrate such additional functionality, we can take the following methodology:

1. A semantic ontology could be developed to distinguish between prefabricated components and other components that are to be 3D printed, which will be separated by a preprocessor.

2. A scheduling algorithm can then be developed for generating the dependencies of all of the prefabricated components of a part, taking into account the unique constraints in the assembly process, such as the size, shape, orientation, material deformation, storage location, and destination.

3. A motion planning algorithm then needs to be developed for the digital assembly robots to precisely pick and place the prefabricated components considering the constraints from other levels of operation.

4. A postprocessor can then be used to merge the planned motion for the digital assembly robots into the DDT of the to-be-printed model. One challenge is that assembly is a

discrete manufacturing operation (e.g., move, pick, place, etc.) while printing is a continuous motion. To address this challenge, we can use cooperation at a lower level, i.e., lower the level of granularity (e.g., in layers in Figure 1.3 in Chapter 1) and develop a set of criteria for merging the discrete assembly motion with the discretized printing process based on the location of the prefabricated components.

Bibliography

[1] N. R. Council *et al.*, *Visionary manufacturing challenges for 2020*. National Academies Press, 1998.

[2] K. Ishii, C. Juengel, and C. F. Eubanks, "Design for product variety: key to product line structuring," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 17179. American Society of Mechanical Engineers, 1995, pp. 499–506.

[3] R. B. Stake, "A hierarchical classification of the reasons for dividing products into modules: a theoretical analysis of module drivers," Ph.D. dissertation, Institutionen för produktionssystem, 1999.

[4] A. De Toni and S. Tonchia, "Manufacturing flexibility: a literature review," *International journal of production research*, vol. 36, no. 6, pp. 1587–1617, 1998.

[5] H. A. ElMaraghy, "Flexible and reconfigurable manufacturing systems paradigms," *International journal of flexible manufacturing systems*, vol. 17, no. 4, pp. 261–276, 2005.

[6] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, and H. Van Brussel, "Reconfigurable manufacturing systems," *CIRP annals*, vol. 48, no. 2, pp. 527–540, 1999.

[7] R. B. Aronson, "Operation plug-and-play is on the way," *Manufacturing Engineering*, vol. 118, no. 3, p. 108, 1997.

[8] J. Siderska and K. S. Jadaan, "Cloud manufacturing: a service-oriented manufacturing paradigm. a review paper," *Engineering Management in Production and Services*, vol. 10, no. 1, 2018.

[9] J. Werfel, K. Petersen, and R. Nagpal, "Designing collective behavior in a termite-inspired robot construction team," *Science*, vol. 343, no. 6172, pp. 754–758, 2014.

[10] M. Shimizu and A. Ishiguro, "An amoeboid modular robot that exhibits real-time adaptive reconfiguration," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 1496–1501.

[11] F. Mondada, L. M. Gambardella, D. Floreano, S. Nolfi, J.-L. Deneuborg, and M. Dorigo, "The cooperation of swarm-bots: Physical interactions in collective robotics," *IEEE Robotics & Automation Magazine*, vol. 12, no. 2, pp. 21–28, 2005.

[12] M. Dorigo, V. Trianni, E. Şahin, R. Groß, T. H. Labella, G. Baldassarre, S. Nolfi, J.-L. Deneubourg, F. Mondada, D. Floreano *et al.*, "Evolving self-organizing behaviors for a swarm-bot," *Autonomous Robots*, vol. 17, no. 2, pp. 223–245, 2004.

[13] J. Werfel and R. Nagpal, "Three-dimensional construction with mobile robots and modular blocks," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 463–479, 2008.

[14] J. K. Werfel, K. Petersen, and R. Nagpal, "Distributed multi-robot algorithms for the termes 3d collective construction system," in *Proceedings of Robotics: Science and Systems*. Institute of Electrical and Electronics Engineers, 2011.

[15] R. Sindhwani and V. Malhotra, "A framework to enhance agile manufacturing system: a total interpretive structural modelling (tism) approach," *Benchmarking: An International Journal*, 2017.

[16] B. Maskell, "The age of agile manufacturing," *Supply Chain Management: An International Journal*, 2001.

[17] F. Sahin, "Manufacturing competitiveness: Different systems to achieve the same results," *Production and Inventory Management Journal*, vol. 41, no. 1, p. 56, 2000.

[18] S. L. Goldman, "Agile competitors and virtual organizations," *Strategies for enriching the customer*, 1995.

[19] J. Browne, D. Dubois, K. Rathmill, S. P. Sethi, K. E. Stecke *et al.*, "Classification of flexible manufacturing systems," *The FMS magazine*, vol. 2, no. 2, pp. 114–117, 1984.

[20] A. M. El-Tamimi, M. H. Abidi, S. H. Mian, and J. Aalam, "Analysis of performance measures of flexible manufacturing system," *Journal of King Saud University-Engineering Sciences*, vol. 24, no. 2, pp. 115–129, 2012.

[21] D. Gupta and J. A. Buzacott, "A framework for understanding flexibility of manufacturing systems," *Journal of manufacturing systems*, vol. 8, no. 2, pp. 89–97, 1989.

[22] Y. Koren, X. Gu, and W. Guo, "Reconfigurable manufacturing systems: Principles, design, and future trends," *Frontiers of Mechanical Engineering*, vol. 13, no. 2, pp. 121–136, 2018.

[23] M. Yuan, K. Deng, and W. A. Chaovalitwongse, "Manufacturing resource modeling for cloud manufacturing," *International Journal of Intelligent Systems*, vol. 32, no. 4, pp. 414–436, 2017.

[24] J. S. Srai, M. Kumar, G. Graham, W. Phillips, J. Tooze, S. Ford, P. Beecher, B. Raj, M. Gregory, M. K. Tiwari *et al.*, "Distributed manufacturing: scope, challenges and opportunities," *International Journal of Production Research*, vol. 54, no. 23, pp. 6917–6935, 2016.

[25] E. Rauch, M. Dallinger, P. Dallasega, and D. T. Matt, "Sustainability in manufacturing through distributed manufacturing systems (dms)," *Procedia CIRP*, vol. 29, pp. 544–549, 2015.

[26] C. Kohtala, "Addressing sustainability in research on distributed production: an integrated literature review," *Journal of Cleaner Production*, vol. 106, pp. 654–668, 2015.

[27] X. Zhang, M. Li, J. H. Lim, Y. Weng, Y. W. D. Tay, H. Pham, and Q.-C. Pham, "Large-scale 3d printing by a team of mobile robots," *Automation in Construction*, vol. 95, pp. 98–106, 2018.

[28] Y. Jin, H. A. Pierson, and H. Liao, "Toolpath allocation and scheduling for concurrent fused filament fabrication with multiple extruders," *IISE Transactions*, vol. 51, no. 2, pp. 192–208, 2019.

[29] L. J. Love, "Utility of big area additive manufacturing (baam) for the rapid manufacture of customized electric vehicles," Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), Tech. Rep., 2015.

[30] H. Shen, L. Pan, and J. Qian, "Research on large-scale additive manufacturing based on multi-robot collaboration technology," *Additive Manufacturing*, vol. 30, p. 100906, 2019.

[31] D. S. Hermann and R. Larson, "Selective mask sintering for rapid production of parts, implemented by digital printing of optical toner masks," in *NIP & digital fabrication conference*, vol. 2008, no. 2. Society for Imaging Science and Technology, 2008, pp. 885–889.

[32] N. Oxman, J. Duro-Royo, S. Keating, B. Peters, and E. Tsai, "Towards robotic swarm printing," *Architectural Design*, vol. 84, no. 3, pp. 108–115, 2014.

[33] B. P. Gerkey and M. J. Mataric, "Multi-robot task allocation: Analyzing the complexity and optimality of key architectures," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 3. IEEE, 2003, pp. 3862–3868.

[34] Y. Zhang and L. E. Parker, "Considering inter-task resource constraints in task allocation," *Autonomous Agents and Multi-Agent Systems*, vol. 26, no. 3, pp. 389–419, 2013.

[35] M. Gini, "Multi-robot allocation of tasks with temporal and ordering constraints," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[36] C. Artigues, "On the strength of time-indexed formulations for the resource-constrained project scheduling problem," *Operations Research Letters*, vol. 45, no. 2, pp. 154–159, 2017.

[37] Y.-K. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Computing Surveys (CSUR)*, vol. 31, no. 4, pp. 406–471, 1999.

[38] K. E. Booth, "Optimization approaches to multi-robot planning and scheduling," in *The 26th International Conference on Automated Planning and Scheduling, London, UK, June*, 2016, pp. 12–17.

[39] Z. Wang and M. Gombolay, "Learning scheduling policies for multi-robot coordination with graph attention networks," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4509–4516, 2020.

[40] N. Atay and B. Bayazit, "Mixed-integer linear programming solution to multi-robot task allocation problem," 2006.

[41] K. Sundar and S. Rathinam, "Route planning algorithms for unmanned aerial vehicles with refueling constraints," in *2012 American Control Conference (ACC)*. IEEE, 2012, pp. 3266–3271.

[42] A. Khuntia, B. Choudhury, and B. Biswal, "An optimized task allocation for multi robot systems using soft computing techniques," in *2012 National Conference on Computing and Communication Systems*. IEEE, 2012, pp. 1–6.

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

[44] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Cooperative Robots and Sensor Networks 2015*, pp. 31–51, 2015.

[45] K. Padmanabhan Panchu, M. Rajmohan, R. Sundar, and R. Baskaran, "Multi-objective optimisation of multi-robot task allocation with precedence constraints." *Defence Science Journal*, vol. 68, no. 2, 2018.

[46] E. G. Jones, M. B. Dias, and A. Stentz, "Time-extended multi-robot coordination for domains with intra-path constraints," *Autonomous robots*, vol. 30, no. 1, pp. 41–56, 2011.

[47] C. Wei, Z. Ji, and B. Cai, "Particle swarm optimization for cooperative multi-robot task allocation: a multi-objective approach," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2530–2537, 2020.

[48] S. Sivanandam, P. Visalakshi, and A. Bhuvaneswari, "Multiprocessor scheduling using hybrid particle swarm optimization with dynamically varying inertia." *Int. J. Comput. Sci. Appl.*, vol. 4, no. 3, pp. 95–106, 2007.

[49] Z. Xu, F. Xia, and X. Zhang, "Multi-robot dynamic task allocation using modified ant colony system," in *International Conference on Artificial Intelligence and Computational Intelligence*. Springer, 2009, pp. 288–297.

[50] N. Gunantara and I. Nurweda Putra, "The characteristics of metaheuristic method in selection of path pairs on multicriteria ad hoc networks," *Journal of Computer Networks and Communications*, vol. 2019, 2019.

[51] T. Schmickl, R. Thenius, C. Moeslinger, G. Radspieler, S. Kernbach, M. Szymanski, and K. Crailsheim, "Get in touch: cooperative decision making based on robot-to-robot collisions," *Autonomous Agents and Multi-Agent Systems*, vol. 18, no. 1, pp. 133–155, 2009.

[52] L. Jin, C. Zhang, and X. Shao, "An effective hybrid honey bee mating optimization algorithm for integrated process planning and scheduling problems," *The International Journal of Advanced Manufacturing Technology*, vol. 80, no. 5, pp. 1253–1264, 2015.

[53] G. Gong, Q. Deng, R. Chiong, X. Gong, H. Huang, and W. Han, "Remanufacturing-oriented process planning and scheduling: mathematical modelling and evolutionary optimisation," *International Journal of Production Research*, vol. 58, no. 12, pp. 3781–3799, 2020.

[54] L. Meng, C. Zhang, X. Shao, and Y. Ren, "Milp models for energy-aware flexible job shop scheduling problem," *Journal of cleaner production*, vol. 210, pp. 710–723, 2019.

[55] J. Zhang, G. Ding, Y. Zou, S. Qin, and J. Fu, "Review of job shop scheduling research and its new perspectives under industry 4.0," *Journal of Intelligent Manufacturing*, vol. 30, no. 4, pp. 1809–1830, 2019.

[56] V. Tereshchuk, J. Stewart, N. Bykov, S. Pedigo, S. Devasia, and A. G. Banerjee, "An efficient scheduling algorithm for multi-robot task allocation in assembling aircraft structures," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3844–3851, 2019.

[57] M. Petrović, N. Vuković, M. Mitić, and Z. Miljković, "Integration of process planning and scheduling using chaotic particle swarm optimization algorithm," *Expert systems with Applications*, vol. 64, pp. 569–588, 2016.

[58] J. McPherson and W. Zhou, "A chunk-based slicer for cooperative 3d printing," *Rapid Prototyping Journal*, 2018.

[59] L. Poudel, Z. Sha, and W. Zhou, "Mechanical strength of chunk-based printed parts for cooperative 3d printing," *Procedia Manufacturing*, vol. 26, pp. 962–972, 2018.

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

[61] E. W. Weisstein, "Acyclic digraph," *https://mathworld. wolfram. com/*, 2003.

[62] S. Elagandula, L. Poudel, Z. Sha, and W. Zhou, "Multi-robot path planning for cooperative 3d printing," in *International Manufacturing Science and Engineering Conference*, vol. 84256. American Society of Mechanical Engineers, 2020, p. V001T01A034.

[63] B. T. Wittbrodt, A. G. Glover, J. Laureto, G. Anzalone, D. Oppliger, J. Irwin, and J. M. Pearce, "Life-cycle economic analysis of distributed manufacturing with open-source 3-d printers," *Mechatronics*, vol. 23, no. 6, pp. 713–726, 2013.

[64] P. M. Bhatt, R. K. Malhan, A. V. Shembekar, Y. J. Yoon, and S. K. Gupta, "Expanding capabilities of additive manufacturing through use of robotics technologies: A survey," *Additive manufacturing*, vol. 31, p. 100933, 2020.

[65] W. Wan, K. Harada, and K. Nagata, "Assembly sequence planning for motion planning," *Assembly Automation*, 2018.

[66] S. Shriyam and S. K. Gupta, "Task assignment and scheduling for mobile robot teams," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 51807. American Society of Mechanical Engineers, 2018, p. V05AT07A075.

[67] A. V. Aho, M. R. Garey, and J. D. Ullman, "The transitive reduction of a directed graph," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 131–137, 1972.

[68] L. Poudel, W. Zhou, and Z. Sha, "A generative approach for scheduling multi-robot cooperative three-dimensional printing," *Journal of Computing and Information Science in Engineering*, vol. 20, no. 6, p. 061011, 2020.

[69] R. Schirru, A. Martinez, C. Pereira, R. Domingos, M. Machado, and L. Machado, "Intelligent soft computing in nuclear engineering in brazil," *Progress in Nuclear Energy*, vol. 35, no. 3-4, pp. 367–391, 1999.

[70] N. M. Razali, J. Geraghty *et al.*, "Genetic algorithm performance with different selection strategies in solving tsp," in *Proceedings of the world congress on engineering*, vol. 2, no. 1. International Association of Engineers Hong Kong, 2011, pp. 1–6.

[71] A. Lipowski and D. Lipowska, "Roulette-wheel selection via stochastic acceptance," *Physica A: Statistical Mechanics and its Applications*, vol. 391, no. 6, pp. 2193–2196, 2012.

[72] C. R. Reeves, "A genetic algorithm for flowshop sequencing," *Computers & operations research*, vol. 22, no. 1, pp. 5–13, 1995.

[73] J. Yu, "Intractability of optimal multirobot path planning on planar graphs," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 33–40, 2015.

[74] M. Darrah, W. Niland, and B. Stolarik, "Multiple uav dynamic task allocation using mixed integer linear programming in a sead mission," in *Infotech@ Aerospace*, 2005, p. 7164.

[75] C. Sarkar, H. S. Paul, and A. Pal, "A scalable multi-robot task allocation algorithm," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5022–5027.

[76] F. Zitouni, R. Maamri, and S. Harous, "Fa–qabc–mrta: a solution for solving the multi-robot task allocation problem," *Intelligent Service Robotics*, vol. 12, no. 4, pp. 407–418, 2019.

[77] Q. Zhang, H. Manier, and M.-A. Manier, "A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times," *Computers & Operations Research*, vol. 39, no. 7, pp. 1713–1723, 2012.

[78] M. Gendreau, M. Iori, G. Laporte, and S. Martello, "A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints," *Networks: An International Journal*, vol. 51, no. 1, pp. 4–18, 2008.

[79] S. Thabit and A. Mohades, "Multi-robot path planning based on multi-objective particle swarm optimization," *IEEE Access*, vol. 7, pp. 2138–2147, 2018.

[80] S. Lorpunmanee, M. N. Sap, A. H. Abdullah, and C. Chompoo-inwai, "An ant colony optimization for dynamic job scheduling in grid environment," *International Journal of Computer and Information Science and Engineering*, vol. 1, no. 4, pp. 207–214, 2007.

[81] G. Sánchez-Ante, F. Ramos, and J. Frausto, "Cooperative simulated annealing for path planning in multi-robot systems," in *Mexican International Conference on Artificial Intelligence*. Springer, 2000, pp. 148–157.

[82] K. Balan and C. Luo, "Optimal trajectory planning for multiple waypoint path planning using tabu search," in *2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, 2018, pp. 497–501.

[83] L. Jaillet and J. M. Porta, "Path planning with loop closure constraints using an atlas-based rrt," in *Robotics Research*. Springer, 2017, pp. 345–362.

[84] E. Lejeune, S. Sarkar, and L. Jezequel, "A survey of the multi-agent pathfinding problem," 2021.

[85] G. Sartoretti, Y. Wu, W. Paivine, T. S. Kumar, S. Koenig, and H. Choset, "Distributed reinforcement learning for multi-robot decentralized collective construction," in *Distributed autonomous robotic systems*. Springer, 2019, pp. 35–49.

[86] C. L. Ortiz, R. Vincent, and B. Morisset, "Task inference and distributed task management in the centibots robotic system," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 2005, pp. 860–867.

[87] J. Peres, P. F. F. Rosa, and R. Choren, "A multi-agent architecture for swarm robotics systems," in *2017 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*. IEEE, 2017, pp. 130–135.

[88] M. Badreldin, A. Hussein, and A. Khamis, "A comparative study between optimization and market-based approaches to multi-robot task allocation." *Advances in Artificial Intelligence (16877470)*, 2013.

[89] K. Hildebrand, B. Bickel, and M. Alexa, "Orthogonal slicing for additive manufacturing," *Computers & Graphics*, vol. 37, no. 6, pp. 669–675, 2013.

[90] L. Poudel, L. G. Marques, R. A. Williams, Z. Hyden, P. Guerra, O. L. Fowler, S. J. Moquin, Z. Sha, and W. Zhou, "Architecting the cooperative 3d printing system," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 83983. American Society of Mechanical Engineers, 2020, p. V009T09A029.

[91] X. Chen, A. Golovinskiy, and T. Funkhouser, "A benchmark for 3d mesh segmentation," *Acm transactions on graphics (tog)*, vol. 28, no. 3, pp. 1–12, 2009.

[92] A. Shamir, "A survey on mesh segmentation techniques," in *Computer graphics forum*, vol. 27, no. 6. Wiley Online Library, 2008, pp. 1539–1556.

[93] L. Luo, I. Baran, S. Rusinkiewicz, and W. Matusik, "Chopper: Partitioning models into 3d-printable parts," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, pp. 1–9, 2012.

[94] R. Hu, H. Li, H. Zhang, and D. Cohen-Or, "Approximate pyramidal shape decomposition." *ACM Trans. Graph.*, vol. 33, no. 6, pp. 213–1, 2014.

[95] J. Vanek, J. G. Galicia, B. Benes, R. Měch, N. Carr, O. Stava, and G. Miller, "Packmerger: A 3d print volume optimizer," in *Computer Graphics Forum*, vol. 33, no. 6. Wiley Online Library, 2014, pp. 322–332.

[96] H. Medellín, J. Corney, J. B. C. Davies, T. Lim, and J. M. Ritchie, "Algorithms for the physical rendering and assembly of octree models," *Computer-Aided Design*, vol. 38, no. 1, pp. 69–85, 2006.

[97] M. Yao, Z. Chen, L. Luo, R. Wang, and H. Wang, "Level-set-based partitioning and packing optimization of a printable model," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, pp. 1–11, 2015.

[98] P. Song, Z. Fu, L. Liu, and C.-W. Fu, "Printing 3d objects with interlocking parts," *Computer Aided Geometric Design*, vol. 35, pp. 137–148, 2015.

[99] S.-H. Ahn, M. Montero, D. Odell, S. Roundy, and P. K. Wright, "Anisotropic material properties of fused deposition modeling abs," *Rapid prototyping journal*, 2002.

[100] H. Rezayat, W. Zhou, A. Siriruk, D. Penumadu, and S. Babu, "Structure–mechanical property relationship in fused deposition modelling," *Materials Science and Technology*, vol. 31, no. 8, pp. 895–903, 2015.

[101] J. Chacón, M. A. Caminero, E. García-Plaza, and P. J. Núnez, "Additive manufacturing of pla structures using fused deposition modelling: Effect of process parameters on mechanical properties and their optimal selection," *Materials & Design*, vol. 124, pp. 143–157, 2017.

[102] H. Kim, E. Park, S. Kim, B. Park, N. Kim, and S. Lee, "Experimental study on mechanical properties of single-and dual-material 3d printed products," *Procedia Manufacturing*, vol. 10, pp. 887–897, 2017.

[103] D. Espalin, J. A. Ramirez, F. Medina, and R. Wicker, "Multi-material, multi-technology fdm: exploring build process variations," *Rapid Prototyping Journal*, 2014.

[104] S. G. Subramanian, M. Eng, V. R. Krishnamurthy, and E. Akleman, "Delaunay lofts: A biologically inspired approach for modeling space filling modular structures," *Computers & Graphics*, vol. 82, pp. 73–83, 2019.

Appendix

A All Publications Published, Submitted, and Planned

1. L. Poudel, C. Blair, J. McPherson, Z. Sha, and W. Zhou, "A Heuristic Scaling Strategy for Multi-Robot Cooperative Three-Dimensional Printing," J. Comput. Inf. Sci. Eng., 2020, doi: 10.1115/1.4045143.

2. L. Poudel, W. Zhou, and Z. Sha, "A Generative Approach for Scheduling Multi-Robot Cooperative Three-Dimensional Printing," J. Comput. Inf. Sci. Eng., 2020, doi: 10.1115/1.4047261.

3. L. Poudel, W. Zhou, and Z. Sha, "Resource-Constrained Scheduling for Multi-Robot Cooperative Three-Dimensional Printing," J. Mech. Des., vol. 143, no. 7, p. 072002, 2021.

4. Z. Zhang, L. Poudel, Z. Sha, W. Zhou, and D. Wu, "Data-driven predictive modeling of tensile behavior of parts fabricated by cooperative 3d printing," J. Comput. Inf. Sci. Eng., vol. 20, no. 2, p. 021002, 2020.

5. L. Poudel, W. Zhou, and Z. Sha, "Towards Swarm Manufacturing: Architecting A Cooperative 3D Printing System," J. Mech. Des., 2021. (In review)

6. L. Poudel, W. Zhou, and Z. Sha, "Decentralized Approach to Cooperative 3D Printing Planning," J. Comput. Inf. Sci. Eng,. 2021. (In progress)

7. L. Poudel, W. Zhou, and Z. Sha, "Computational design of scheduling strategies for multi-robot cooperative 3D printing," in Proceedings of the ASME Design Engineering Technical Conference, 2019, vol. 1. doi: 10.1115/DETC2019-97640.

8. L. Poudel, Z. Sha, and W. Zhou, "Mechanical strength of chunk-based printed parts for cooperative 3D printing," in Procedia Manufacturing, Bryan, TX, 2018, vol. 26, pp. 962–972. doi: 10.1016/j.promfg.2018.07.123.

9. L. Poudel et al., "Architecting the Cooperative 3D Printing System," in International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 2020, vol. 83983, p. V009T09A029.

10. S. Elagandula, L. Poudel, Z. Sha, and W. Zhou, "Multi-Robot Path Planning for Cooperative 3D Printing," in International Manufacturing Science and Engineering Conference, 2020, vol. 84256, p. V001T01A034.

11. S. Elagandula, L. Poudel, Z. Sha, and W. Zhou, "Enabling Multi-Robot Cooperative Additive Manufacturing: Centralized vs. Decentralized Approaches," in International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 2021. (Accepted)