# Decentralized and Centralized Planning for Multi-Robot Additive Manufacturing

**Laxmi Poudel**
Department of Mechanical Engineering,
University of Michigan,
Ann Arbor, MI 48109
e-mail: lpoudel@umich.edu

**Saivipulteja Elagandula**
Department of Computer Science,
and Computer Engineering,
University of Arkansas,
Fayetteville, AR 72701
e-mail: selagand@uark.edu

**Wenchao Zhou**
Department of Mechanical Engineering,
University of Arkansas,
Fayetteville, AR 72701
e-mail: zhouw@uark.edu

**Zhenghui Sha**[1]
Department of Mechanical Engineering,
University of Texas,
Austin, TX 78712
e-mail: zsha@austin.utexas.edu

*In this paper, we present a decentralized approach based on a simple set of rules to schedule multi-robot cooperative additive manufacturing (AM). The results obtained using the decentralized approach are compared with those obtained from an optimization-based method, representing the class of centralized approaches for manufacturing scheduling. Two simulated case studies are conducted to evaluate the performance of both approaches in total makespan. In the first case, four rectangular bars of different dimensions from small to large are printed. Each bar is first divided into small subtasks (called chunks), and four robots are then assigned to cooperatively print the resulting chunks. The second case study focuses on testing geometric complexity, where four robots are used to print a mask stencil (an inverse stencil, not face covering). The result shows that the centralized approach provides a better solution (shorter makespan) compared to the decentralized approach for small-scale problems (i.e., a few robots and chunks). However, the gap between the solutions shrinks while the scale increases, and the decentralized approach outperforms the centralized approach for large-scale problems. Additionally, the runtime for the centralized approach increased by 39-fold for the extra-large problem (600 chunks and four robots) compared to the small-scale problem (20 chunks and four robots). In contrast, the runtime for the decentralized approach was not affected by the scale of the problem. Finally, a Monte-Carlo analysis was performed to evaluate the robustness of the centralized approach against uncertainties in AM. The result shows that the variations in the printing time of different robots can lead to a significant discrepancy between the generated plan and the actual implementation, thereby causing collisions between robots that should have not happened if there were no uncertainties. On the other hand, the decentralized approach is more robust because a collision-free schedule is generated in real-time.*
[DOI: 10.1115/1.4055735]

*Keywords: design for additive manufacturing, multi-robot planning, additive manufacturing, cooperative 3D printing*

## 1 Introduction

Cooperative 3D printing (C3DP), as illustrated in Fig. 1(*a*), is an emerging additive manufacturing (AM) technology that uses multiple mobile 3D printing robots to accomplish large-scale printing tasks. Chunked-based printing [1], which is illustrated in Fig. 1(*b*) using fused filament deposition (FDM) [2] technology, is one of the manifestations of multi-robot cooperative 3D printing, where a part is first divided into multiple chunks (a large part when geometrically partitioned results in smaller printing tasks, called chunks), and the chunks are then assigned to multiple robots to print simultaneously, thus reducing printing time and increasing the printing scale. The efficiency of cooperative 3D printing requires careful coordination among the robots, which requires them to work in parallel when possible and avoid collision with other robots and previously printed materials. So the constraints in an environment are changing both in space and time.

There are generally three approaches to solve 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. Centralized approaches often involve mathematical optimization, such as linear programming [3], integer programming [4], and combinatorial optimization [5]. These optimization approaches use the branch and bound, branch and cut method to converge to optimal results [6]. Metaheuristics are also often used in centralized planning, such as simulated annealing [7] and genetic algorithm (GA) [8], 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 [9]. 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 main differences between the two approaches are highlighted in Table 1. While both centralized and decentralized approaches have been widely studied in the multi-robot systems literature over the past decades, the differentiation between the two approaches in multi-robot cooperative manufacturing is not quite pronounced. Actually, the use of decentralized approaches for cooperative AM has not been reported in the existing literature to the best of our knowledge.

In our previous study [10], we implemented two centralized approaches (modified genetic algorithm (MGA) and mixed-integer linear programming (MILP)) to solve the multi-robot coordination for C3DP. The centralized approaches can provide satisfactory control of the system because the status of all the robots is always known at any given time. Using the approaches, we obtained near-optimal solutions for both small-scale and large-scale problems (the scale refers to the number of robots and chunks). 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 over a large area, where many mobile robots have to travel for a long distance on the factory

[1]Corresponding author.

Contributed by the Design for Manufacturing Committee of ASME for publication in the Journal of Mechanical Design. Manuscript received April 2, 2022; final manuscript received September 15, 2022; published online October 10, 2022. Assoc. Editor: Deepesh Toshniwal.

  

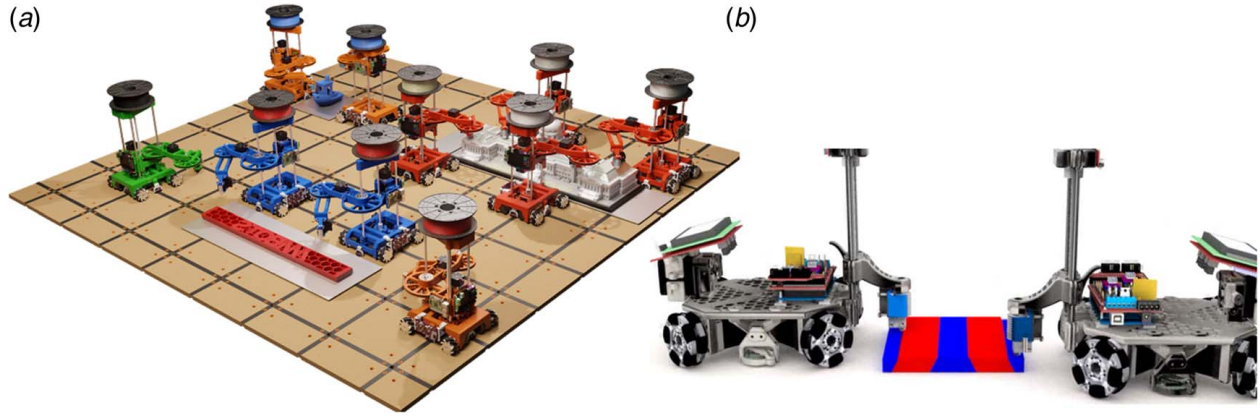


**Fig. 1 (*a*) Demonstration of multi-robot cooperative 3D printing, where multiple jobs are simultaneously printed using multiple robots and (*b*) demonstration of chunk-based printing, where a part is discretized into smaller chunks so that it can be printed using two robots [36]**

**Table 1 Differences between the centralized approaches versus 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 the 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 |

floor to accomplish multiple print jobs. The planning (the term planning is used to indicate both scheduling and path planning in this paper) of robots in those situations becomes complicated due to the increase of dependencies among robots for multiple print tasks.

Therefore, several research questions motivated us to study decentralized approaches to realizing multi-robot cooperative manufacturing. For example, while the centralized approach has been proven effective for small-scale problems, could it be a bottleneck in large-format C3DP? The centralized approach requires a robust communication scheme between the central planner and the entire team. Can that still be established at a larger scale with high reliability and reasonably low cost? In addition, as the number of robots increases, uncertainties in executions increase because it is difficult to predetermine 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. All these problems can 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 might not provide a theoretical guarantee for optimal global solutions and may often be far from optimal in both path planning and scheduling, could it be a feasible solution if it demands no expensive communication? Motivated by answering these research questions, we aim to investigate the application of the two paradigms in both large and small-scale fabrication to understand the strength and weaknesses of each paradigm in C3DP. These questions also motivate us to explore and develop a decentralized multi-robot planning method in cooperative manufacturing and compare its performance with a centralized approach.

In this paper, 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 order while driving. Each agent (robot) adheres to these rules and coordinates based on the local exchange of information. These agents are unaware of the global information, and the framework does not need a central planner to assign tasks and coordinate the path planning. The agents can only share information with nearby agents when they are in proximity to one another. They then use the newly received information to avoid conflict, such as collisions while traveling, and determine where the next print can be done. The results of this decentralized planner are compared with that of a centralized multi-robot planner, which was presented in our previous study [10]. The centralized planner uses a modified genetic algorithm to assign and schedule print jobs to individual agents. For the path planning between work stations, an A* search algorithm is used to obtain collision-free paths for the generated schedule. The comparison is based on multiple criteria, such as scalability, computational time, and uncertainty (e.g., robot failure). The rest of the paper is organized as follows. The relevant works in the existing literature are reviewed in Sec. 2, followed by a detailed introduction to the decentralized approach in Sec. 3. The comparison between the decentralized and centralized approaches is presented in Sec. 4, followed by a discussion and interpretation of the results in Sec. 5. Finally, the conclusion and the future work are presented in Sec. 6.

## 2 Relevant Research

Multi-robot task planning has seen applications of both centralized and decentralized approaches. Traditionally, the centralized

approaches have dominated the multi-robot task planning problems, while more recently, an increasing number of decentralized and distributed approaches have been researched. However, the application of decentralized approaches in cooperative AM with multiple robots is still rare.

**2.1 Centralized Approaches to Multi-Robot Planning.** The use of a centralized approach for multi-robot planning is abundant in the literature. Though multi-robot planning includes multi-robot task allocation (MRTA) and multi-agent pathfinding (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 [11,12]. The centralized approaches to MRTA include optimization-based approaches [13,14]. Atay and Bayazit used the MILP approach to allocate heterogeneous robots to maximize the coverage of the area for the robot's operation [15]. Similarly, Darrah et al. also used MILP to solve the MRTA problem in the context of unmanned ariel vehicles [16]. 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 [17]. In another study, Sarkar et al. presented another heuristics approach called nearest-neighbor-based clustering and routing that scales better compared to other existing state-of-art heuristics ($O(n^3)$) [18]. More recently, Zitouni et al. presented an approach using a two-stage methodology where at the global level, task allocation is done using a firefly algorithm, and local allocation is done by combining quantum genetic algorithm and artificial bee colony optimization [19]. In addition, some other heuristic approaches for MRTA include simulated annealing [7,20], tabu search [21,22], etc.

On the other hand, MAPF has also been studied widely using different centralized approaches. Thabit and Mohades presented a multi-robot path planning approach based on multi-objective (shortness, safety, and smoothness) particle swarm optimization in an unknown environment [23]. Additionally, several heuristic approaches are inspired by the biological system, such as genetic algorithm [24], ant colony optimization [25], particle swarm optimization [17], and have been used to solve path planning problem. Other heuristics include the simulated annealing algorithm [26] and tabu search [27]. 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 a dynamic and unknown environment [28], 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 numerous 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 [29].

**2.2 Decentralized 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 frequent synchronization of execution between multiple robots at a high cost, the execution sequence is largely non-deterministic when the robots are operating independently and unsynchronized. As the number of robots increases, it becomes increasingly difficult to predict the planning outcome over an extended period of time with a centralized approach, and frequent replanning will be needed. 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 environments and without a centralized planner. One widely publicized research using a decentralized approach is conducted by Werfel and Nagpal [30] and Werfel et al. [31]. The authors developed a swarm of termite-inspired multi-robot construction systems solely based on a set of simple rules and local communication between the robots. The system consists of individual robots with minimal 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 [32]. While such an approach demonstrates speed up, it can limit the scalability of the system. Similarly, Ortiz et al. presented the centibots system—a multi-robot distributed system consisting of more than 100 robots in unknown indoor environments for search and rescue problems over extended periods of time [33]. Peres et al. presented a multi-agent swarm robotics architecture to simulate heterogeneous robots that interact with each other and humans to accomplish several types of missions such as surveillance, intruder detection, and leader–follower [34].

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 manufacturing using multiple robots. While some comparative studies exist between the centralized and decentralized approaches [35] in multi-robot systems, the application is limited to discrete tasks such as pick and place, team formation, and warehouse functionalities. However, due to manufacturing constraints, the multi-robot C3DP poses more challenges than common discrete tasks. Thus, this study aims to address the knowledge gap on how centralized and decentralized methods would perform in cooperative manufacturing applications. It does so by presenting a decentralized approach based on a set of rules and comparing the results with those of centralized approaches, such as the modified genetic algorithm method with CBS, through simulation studies.

## 3 Approaches to C3DP Planning

**3.1 The Decentralized Approach for 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 rules and make decisions based on the said rules. We outline these rules in more detail in Sec. 3.1.1. In the subsequent discussion, a job refers to the entire object to be printed. A job is split into chunks, so robots can print portions of the job. An example of chunking is presented in Fig. 6, where a rectangular bar and a Razorback-shaped mask stencil or Razorback mask[2] (an inverse stencil, not face covering, as shown in Fig. 6(*b*)) are chunked into 30 and 24 chunks, respectively. 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 (positive *Y*-axis) and down (negative *Y*-axis) (see Fig. 5 for reference).

[2]Razorback is a special term used to describe the school mascot of the University of Arkansas.

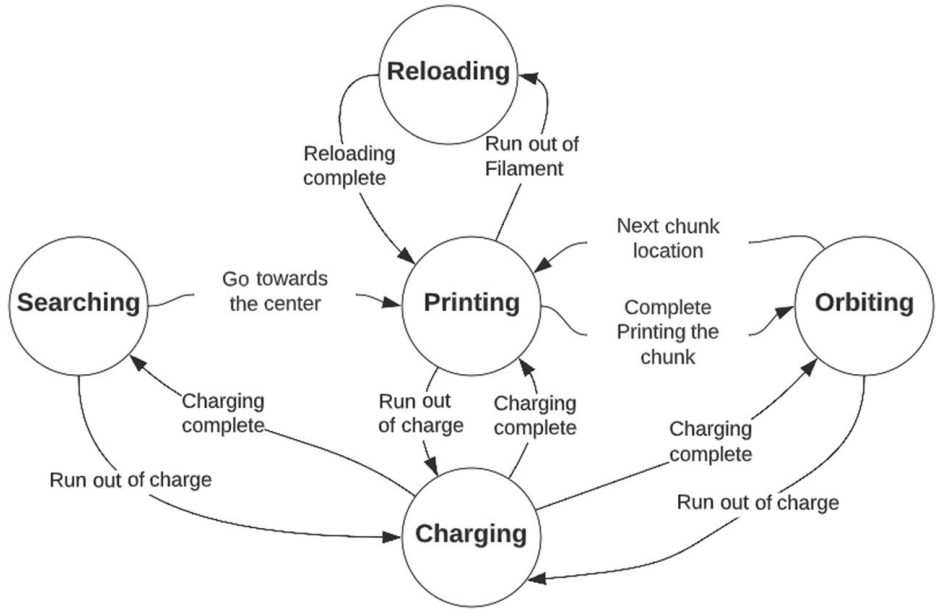


**Fig. 2 Finite state machine for decentralized C3DP workflow**

(3) Robots are set to have a limited communication range to reduce communication costs, 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 can be accessed when they are ready to print individual chunks and
  (b) the location of each chunk to be printed.

These capabilities allow robots to act independently and access sufficient information to decide whether they can print at a certain location or not. In this approach, a job is assumed to be chunked and the placement of the job on the floor is decided beforehand and is not part of the proposed decentralized planning. This information is passed to the robot prior to printing. Robots have five states: *searching, printing, orbiting, charging, and reloading*. The flow state of the printing robots is shown in Fig. 2. They first start with the searching state, where robots move towards the center area of the grid using the coordinates information of chunks. That is where the job is located. Once a robot moves from one grid point to another, 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 a location is found, it determines whether it is allowed to print using the geometric rule. If it can print, it will transition to a printing state and start printing the chunk. On the other hand, during searching, once a robot finds a printed chunk or a chunk is being printed by another robot, it transitions from the searching state (looking for where the job is located in the floor space) to the orbiting state. In its orbiting state, the robot will orbit around the printed chunks and the printing robots following the parallel movement rule. 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 predetermined threshold (e.g., 80%), they will need to be recharged and, therefore, will move toward the charging station located at one of the corners of the floor. Once the robots reach the charging station, they transition to the charging state. The battery of the robot must be fully recharged before it can leave the charging station. Once fully recharged, the robot 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. It will transition to the reloading state and notify the user to reload the filament if that happens. Once the new filament is loaded, it transitions back to the state before running out of filament.

### 3.1.1 *The Set of Rules*

(A) *The rule of geometry:* In our previous study, we 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) [1,36]. This chunking method allows the material of the adjacent chunks to be deposited on the sloped surface of the already printed chunk, creating an instant bond between the two chunks. Since the adjacent chunks have a sloped surface interface with each other, this will create geometric constraints. The rule of geometry is established to ensure geometric dependencies between adjacent chunks are followed. For example, as shown in Fig. 3, chunk 0 and chunk 1 must be printed before chunk 2 because chunk 2 has overhangs that prevent parts of chunks 0 and chunk 1 from being printed. Otherwise, the printing nozzle will collide with the said overhangs of chunk 2 while printing chunk 0 and chunk 1. To avoid such collisions, robots must print chunks that have convex slopes (e.g., chunks 0 and chunk 1) before printing adjacent chunks with concave slopes (e.g., chunk 2). Such geometric dependencies are stored as directed

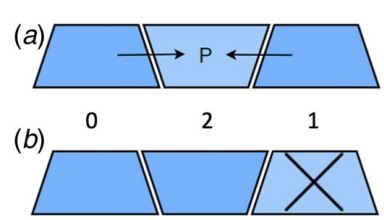


**Fig. 3 (*a*) Rule of geometry indicates that chunk 0 and chunk 1 should be printed before chunk 2 and (*b*) Possible issues result from the lack of geometric rule where chunk 1 cannot be printed if chunk 2 is printed first**

graph data structures, where each node represents a chunk, and an edge between two nodes represents the dependent relationship of a pair of chunks. Such data structure 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 or not. For example, in Fig. 3, if the robot reaches the print location where chunk 2 is to be printed, it will inspect whether chunks 0 and 1 have been already printed or not. If either of them has not been printed, the robot will search for a different chunk to print instead of printing chunk 2.

This rule is important as this allows the print job to be carried out properly and prevents robots from printing chunks that would make printing subsequent chunks impossible. While the provided example uses a sloped surface chunking strategy for demonstration, such dependencies will likely exist in any other future geometric partitioning strategies. Thus, the rule of geometry has a general implication to handle similar geometric 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 at an intersection simultaneously. 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. The process is repeated if both the robots generate the exact same number.

(C) *The rule of orbiting:* The rule of orbiting is designed to limit exploration and bring a more systematic approach to robots' search for a print location. For example, when a robot is in searching mode, it can start randomly moving around the floor until it finds a print location, but such an approach could result in excessive movements. To avoid unnecessary exploration, the robot will, in the beginning, move towards the center of the floor (where seed chunks are located. The location of the print object is predetermined by the user before implementing the planning). 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 Fig. 4. No robot is not allowed to turn back or circle the printed chunks in clockwise movements. This rule is inspired by the work done by Werfel et al. [37].

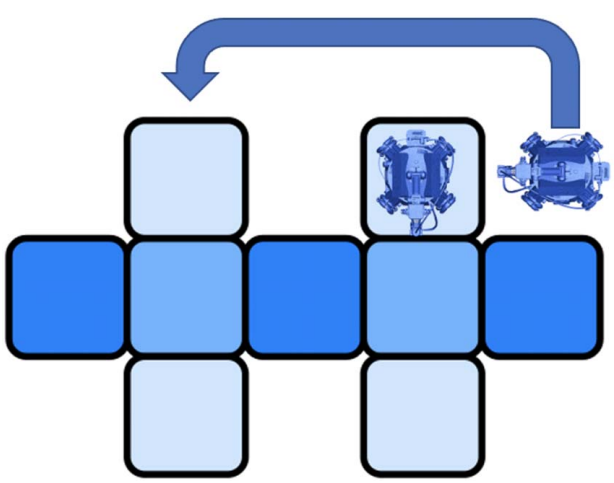

**Fig. 4 Rule of orbiting, where the robot will orbit the printed chunk counterclockwise to find the next print location**

**3.2 The Centralized Approach for C3DP.** Our previous study developed a metaheuristic approach to multi-robot scheduling based on a MGA [10]. The GA is an evolutionary stochastic algorithm widely used in multi robot system 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 to generate print schedules in conjunction with the chunk assignment. Genetic operators are then applied to modify the chunk assignment until a specified number of population generations is achieved. The fitness function of this MGA method is 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. To incorporate collision-free path planning in our previously developed MGA, a CBS method is adopted [38]. 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. A low-level search is done for each node 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 [38]. Once the initial print schedule population is generated in MGA, path planning using CBS is carried out for each chromosome representation of a GA solution. The travel time obtained using CBS is then added to the fitness value of each chromosome using Eq. (1). The equation takes the start time of a chunk, the time it takes a robot to print the chunk, and the time it takes a robot to travel from one location to another to calculate the total time. More details of the equation are provided in Ref. [10]. Thus, a chromosome's overall fitness score includes the total print time and the travel time required for a particular print schedule. This new MGA method enables the generation of solutions to improve 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.

$$T_{\text{total}} = \text{Max}\left(T_{\text{start},ij} + T_{\text{print},ij} + \sum T_{\text{travel},j}\right) \tag{1}$$

where $T_{\text{start},ij}$ is the start time of chunk $i$ on robot $j$, $T_{\text{print},ij}$ is the print time of chunk $i$ on robot $j$, $\sum T_{\text{travel},j}$ is the total travel time of robot $j$ throughout the job, $j = 1, 2, 3, \ldots, m$, robots used for printing, and $i = 1, 2, 3, \ldots, n$, chunks assigned to robot $j$.

## 4 Result

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

**4.1 Simulation Setup.** All the simulations for the comparison are conducted in a custom-designed simulation software programed with PYTHON programming language. The 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

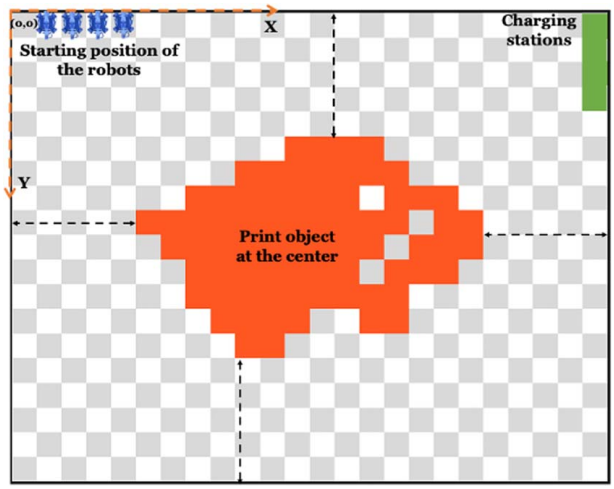


**Fig. 5 Representation of grid world where four robots are ready to print a Razorback mask placed in the center of the grid**

all sides. These five grid points are left to provide enough space for multiple robots to move across simultaneously if needed. A representative visualization is presented in Fig. 5. (2) All the robots, in the beginning, are placed around the origin, as shown in Fig. 5. (3) For decentralized planning, only three states, searching, orbiting, and printing state, are considered in the simulation study. To make it consistent with the centralized planning approach, charging and reloading states are not taken into consideration because they have trivial influence on the overall planning.

**4.2 Evaluation Metrics.** To quantitatively compare the two approaches, we use the makespan of a job as the evaluation metric. This includes both the print time as well as the travel time. The computational complexity of 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 as a representative of decentralized planning and the MGA-CBS approach as a representative of centralized planning as a benchmark for comparison in two different case studies. In the first case study, a simple geometry (a rectangular bar) is divided into multiple chunks and assigned to the robots. 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 in the number of chunks. The second case study uses a more complicated geometry (a Razorback shape). With a more complicated geometry, the size and shape of the chunks vary widely after chunking, as shown in Fig. 6(*b*). The printing of chunks in the case of a rectangular bar can be highly synchronized. For example, chunk 1 and chunk 3 could be printed by robot 1 and robot 2 in Fig. 6(*a*). Since they have similar shapes and sizes, they will be completed together. Once the printing is complete, robot 1 and robot 2 could move to the right and start printing chunk 2 and chunk 4. 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 shorter makespan. However, for the Razorback shape, none of the chunks have similar geometries, and thus, synchronicity cannot be achieved. Therefore, the main purpose of the second case study is to test whether such a large variation in shape and size (and thus, the print time) of chunk geometry has any impact on the performance of the two approaches. Additionally, uncertainties are part of manufacturing processes, more so for additive manufacturing. A Monte-Carlo analysis is conducted to understand how the inherent uncertainties in additive manufacturing (e.g., the discrepancy between the predicted print time and the actual print time) could derail the print planning generated using the centralized approach. In addition to the discrepancy in the print times, other uncertainties include unanticipated nozzle clog, robot failure, etc. Thus, a single value generated for makespan during planning might not represent the actual result during implementation, and statistical measurements such as range and confidence interval should be included. The results, outlining the details of each case, are presented in Sec. 4. 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, a chunking algorithm developed in our previous study [1,39]. The stereolithography (STL) model of the individual chunk is sliced in Ultimaker Cura™ to estimate the print time. The print time obtained from the slicer is used to calculate the simulation time (to simulate printing in a rules-based approach) using Eq. (2). The equation takes the actual print time obtained by slicing the chunk and scales it down to minimize the simulation time. For the sake of consistency, the centralized approach also uses the same unit of time for calculation.

$$\text{Simulation time} = \frac{\sqrt{\dfrac{\text{Print time}}{10}}}{6}\ \text{time-steps} \tag{2}$$

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

### 4.3 Case Studies

*4.3.1 Case I.* The first case study is a rectangular block of dimension $1\ \text{m} \times 0.8\ \text{m} \times 0.015\ \text{m}$ and has a total volume of $0.012\ \text{m}^3$. The rectangular block and the resulting chunks using

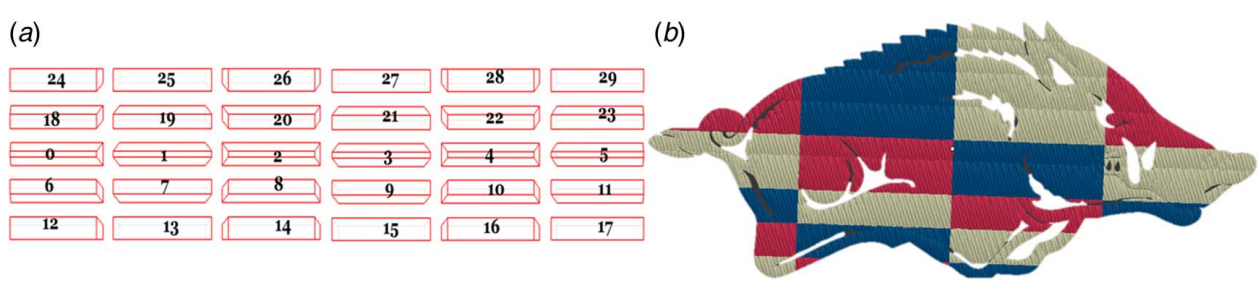


**Fig. 6 (*a*) An exploded view of a chunked rectangular bar and (*b*) a Razorback chunked into multiple heterogeneous chunks [10]**

**Table 2 Data associated with case study I**

| | Small job | Medium job | Large job | Extra-large job |
|---|---|---|---|---|
| Number of chunks | 20 | 100 | 300 | 600 |
| Number of robots | 4 | 4 | 4 | 4 |

the sloped surface chunking strategy are presented in Fig. 6(*a*). Four robots are used to print this job. In order to better understand how both approaches behave with the change of the scale of the problem, we change the scale of this object by increasing its size (resulting in a larger number of chunks) and calculate the result using both approaches. The number of robots is kept constant (four) for different scales of the problem. First, the object is increased by five times, which results in 100 total chunks. Next, we further increased its scale by 15 times (resulting in 300 chunks) and finally by 30 times (resulting in 600 chunks). The summary of the data associated with the first case study is presented in Table 2.

The results obtained using both the centralized and decentralized approaches are presented in Table 3. 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. The number of iterations (max. 100 iterations) and runtime (max. 12 h) (whichever comes first) is used as a stopping criterion for the centralized planner.

*4.3.2 Case II.* The second case study is a Razorback shape with more complicated geometry than a rectangular bar presented in the first case study. A coarse representation of the Razorback shape is presented in Fig. 5 and has the dimension of 610 mm × 267 mm × 52 mm. The Razorback shape was chunked into 50 non-homogeneous chunks, and the printing was simulated using four printing robots.

## 5 Discussion

The charts presented in Fig. 7 illustrate the makespan and average travel time, side by side, of both approaches in case I. Results show that the overall makespan from the centralized approach is better than that from the decentralized approach for small and medium jobs (i.e., 20 and 100 chunks). However, the gap between the makespan obtained using the two approaches decreases as the size of the job increases. As a result, the makespan from the decentralized approach is lower for job sizes with 300 chunks and 600 chunks compared to the makespan from the centralized approach, which can be observed in Fig. 8.

This is likely because, for a combinatorial problem like this, as the number of printing tasks or chunks increases, the search space increases significantly. This has a two-fold impact on the overall solution obtained using the centralized approach. First, as the search space increases, exhaustively searching the solution space becomes impossible, and as a result, a large part of the solution space remains unexplored. Even with a stochastic approach involving GA, it is difficult to search the entire solution space, and thus, the quality of the solution degrades with the increase in the scale of the problem. However, in the decentralized approach, decisions are made based on local information, and the scale of the problem has no apparent effects on the overall quality of the solution. As a result, we see the gap between the makespan between the centralized and decentralized approaches decreasing and eventually reversing as the scale of the problem increases. Second, each iteration for the centralized planner requires longer runtime. This is evident from Table 3, where we can see that the computation time increased by 28 times (for the large job with 300 chunks) to run the same number of iterations (100 iterations), while the scale of the job increased by only 15 times. Additionally, for the extra-large job (600 chunks), the runtime limit (12 h) was reached before the maximum iteration limit (100 iterations). Thus, the result reflects the makespan obtained after only 50 iterations for the extra-large job problem.

In order to reduce the computation runtime, a print schedule could be generated first using the MGA method without coupling with path planning and then 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% on 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 or minutes), this approach might yield acceptable solutions. Thus, while the concurrent scheduling and path planning used in the adopted centralized approach yields a better result, it also demands more computational resources. This is especially true for extremely large-scale problems, as demonstrated by the extra-large job. Meanwhile, the computation time of the decentralized approach did not change much for different scales of the problem, as shown by the data in Table 3. This is because no significant computation is required for planning. As robots start from their initial positions, they make movements based on the rules and instantaneously make decisions without planning for 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 may conclude that the centralized planner is worse than the decentralized planner for a large-scale problem in terms of the overall makespan; however, it is worth noting that this is not always the case. Given enough computational resources and time to converge to an optimal solution, the centralized planning approach may outperform decentralized planning, even for a large-scale problem. The downside is that frequent replanning may be needed if the execution deviates from the plan due to uncertainty and synchronized execution between robots. The data associated with both approaches for the second case study are presented in

**Table 3 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 |

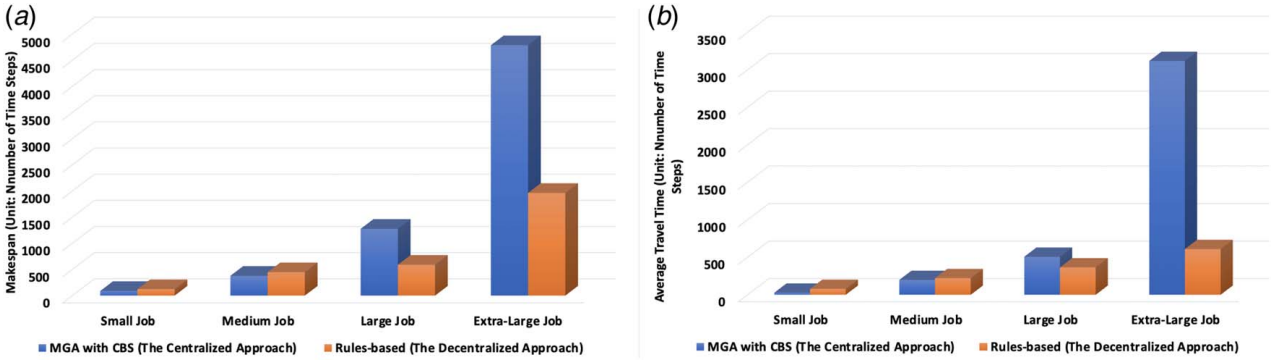


**Fig. 7 The makespan and average travel time by robots in both decentralized and centralized approaches for different scales of the problem in case study I (rectangular bar discretized into multiple chunks, represented by small, medium, large, and extra-large job)**

Table 4. The makespan of both approaches, while not exactly equal, is similar. The results obtained using the centralized approach are marginally better than those obtained using the decentralized approach.

**5.1 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 print time estimate, the actual print time during execution is often different from the estimated time. 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 Fig. 6(*a*)) will be relatively close to the actual print time. On the other hand, the time estimation of more complex geometry (e.g., the Razorback in Fig. 6(*b*)) 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, number of turning points in the printing paths, etc.) and hardware differences (e.g., set value of print speed versus actual speed, the difference of clock frequency between microcontrollers, etc.). 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 uncertainty propagation or cascading effects. As a result of such uncertainties in AM, a makespan value obtained from the planning rarely matches the actual result. We are interested in investigating how such uncertainties impact the actual makespan and comparing the performance of the two approaches when handling uncertainties. Therefore, we conducted a Monte-Carlo simulation to analyze how the uncertainties in the print time affect the print plan for centralized planning.

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 choose Gaussian distribution to simulate the variation in the actual print time. Once a part is divided into multiple chunks, each chunk is sliced using a Cura slicer engine from which an estimated print time is obtained. This print time is used as the mean value to generate the Gaussian 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 as some factor of the mean value. However, due to the lack of proper guides in the existing literature on choosing the value for a standard deviation, one-tenth of the mean value is used. Increasing or decreasing the value of standard deviation changes the spread of the actual makespan. A printing scenario with 60 chunks using four printing robots is used to demonstrate the simulation.

The histogram presented in Fig. 9 shows the wide range of the actual makespan compared to the computed makespan by the centralized planning approach. As a result of uncertainty in the printing time for the individual chunks, the actual makespan can range anywhere from 115 time-steps to 137 time-steps. Such a wide range poses a significant problem in print planning, especially in cases where the printing robots are to be deployed for printing pre-planned chunks at predetermined times. This could result in collisions between the

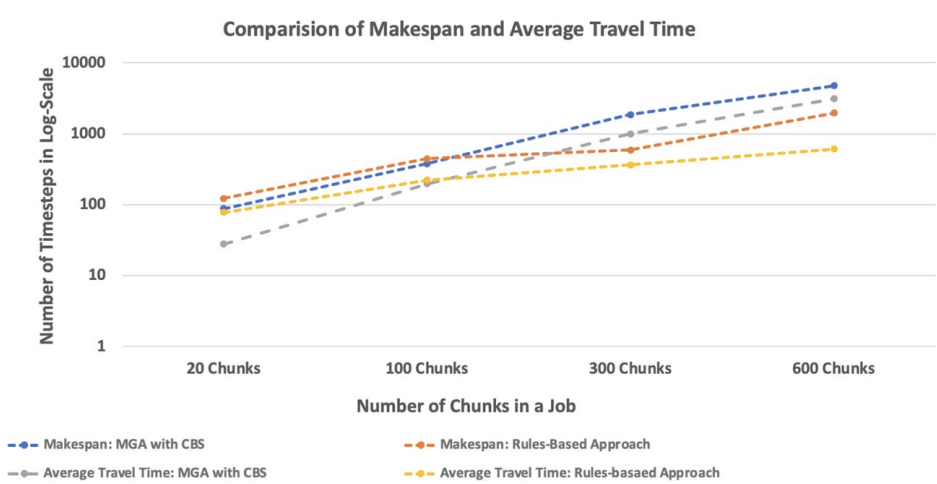


**Fig. 8 The changes in makespan and average travel time obtained using different approaches for different scales of a problem presented for case I, plotted at log-scale**

**Table 4 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 |

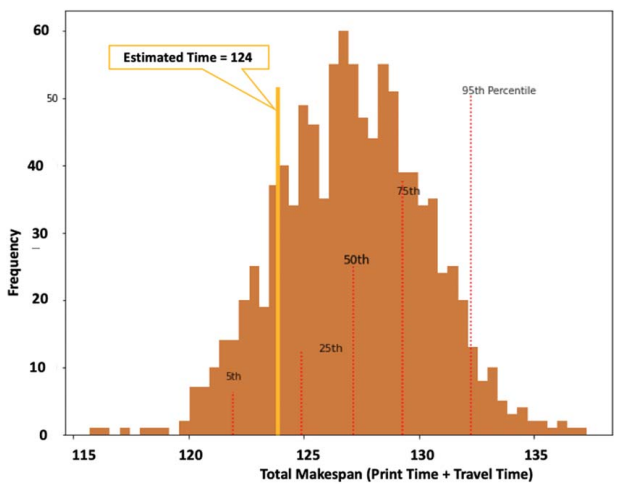


**Fig. 9 Total makespan obtained after running 1000 Monte-Carlo simulation that shows variation in actual print time in the centralized approach. Estimated print time marked solid vertical line for reference.**

robots and the collision between robots and printed parts. Additionally, the calculated value of 124 time-steps is within 25 percentile makespan in the simulated data, i.e., we can say that the actual makespan will be 124 time-steps with only 25% certainty.

While 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 collision-free printing cannot be guaranteed. For example, a chunk $c_{i+1}$ has dependency on chunk $c_i$ (i.e., $c_i$ has to be completed before chunk $c_{i+1}$ can begin) and that robot $R_i$ is scheduled to finish printing $c_i$ at time $t_i$ and robot $R_j$ is assigned to start printing chunk $c_{i+1}$ at $t_i$. However, as a result of uncertainty in the actual print time, robot $R_i$ does not complete printing chunk $c_i$ until time $t_i + \Delta t$ but the robot $R_j$ will start printing chunk $c_{i+1}$ at $t_i$. This violates the dependency or precedence constraints and would result in a collision between the printing robots as well as the robots and the printed part. Thus, additional precautions are required to prevent such accidents, and the optimal or near-optimal schedule that was generated during planning is no longer valid during the actual implementation. As a result, replanning will be needed. 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.

On the other hand, decentralized planning does not suffer from such problems because no pre-planning is required, and the robots make printing decisions based on the local information available to them. Printing robots are provided the dependency or precedence relationship information about the chunks, and the robots print the chunk as they become available (for example, chunk $c_{i+1}$ does not become available until its dependency chunk $c_i$ is finished). Thus, uncertainties do not negatively affect the print process (may cause delay but will not result in a collision) when implemented using the decentralized approach.

## 6 Conclusion and Future Work

In this paper, a rules-based decentralized approach is developed for planning multi-robot cooperative additive manufacturing, the performance of which is compared to an MGA-based centralized approach. 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. On the other hand, the centralized approach 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) and 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 problems. The second case study is a large-scale Razorback shape 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 may indicate that geometric complexity might not have a significant impact on the overall performance of the two approaches; however, a more comprehensive and systematic investigation is needed to make this conclusion conclusive. 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 apparent impact on the functionality of the decentralized approach. Similarly, if a robot fails while traveling, other robots will continue working and take a larger chunk of work to complete the job. A centralized approach, on the other hand, requires replanning of the job. The summary of these differences based on the result and discussion is presented in Table 5.

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

**Table 5 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 |

decentralized approach. However, doing so might require a more computationally taxing algorithm. 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 the centralized approaches (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 the presence of such uncertainties.

Finally, there are no studies that combine exact methods and learning methods for multi-robot cooperative manufacturing problems, because both are computationally demanding approaches. But as computing resources become cheaper and the learning methods become more efficient, one promising approach is imitation learning with exact methods. In that approach, expert demonstration can be done in small-scale problems using exact methods which can be used to learn and implement for large-scale problems. We envision that much research effort towards this direction will be devoted in the near future.

## Acknowledgment

This work is partially supported by the National Science Foundation under Award Number 2112009. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## Conflict of Interest

There are no conflicts of interest.

## Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

## References

[1] McPherson, J., and Zhou, W., 2018, "A Chunk-Based Slicer for Cooperative 3D Printing," Rapid Prototyping J., **24**(9), pp. 1436–1446.
[2] ISO/ASTM, 2015, "ASTM52900-15 Standard Terminology for Additive Manufacturing—General Principles—Terminology," ASTM International, West Conshohocken, PA, 3(4), p. 5.
[3] Artigues, C., 2017, "On the Strength of Time-Indexed Formulations for the Resource-Constrained Project Scheduling Problem," Oper. Res. Lett., **45**(2), pp. 154–159.
[4] Kwok, Y.-K., and Ahmad, I., 1999, "Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors," ACM Comput. Surveys, **31**(4), pp. 406–471.
[5] Booth, K. E. C., 2016, "Optimization Approaches to Multi-Robot Planning and Scheduling," The 26th International Conference on Automated Planning and Scheduling, London, UK, June 12–17, p. 128.
[6] Wang, Z., and Gombolay, M., 2020, "Learning Scheduling Policies for Multi-Robot Coordination With Graph Attention Networks," IEEE Robot. Autom. Lett., **5**(3), pp. 4509–4516.
[7] Khuntia, A., Choudhury, B., and Biswal, B., 2012, "An Optimized Task Allocation for Multi Robot Systems Using Soft Computing Techniques," 2012 National Conference on Computing and Communication Systems, Durgapur, India, Nov. 21–22, IEEE, pp. 1–6.
[8] Shao, X., Li, X., Gao, L., and Zhang, C., 2009, "Integration of Process Planning and Scheduling—A Modified Genetic Algorithm-Based Approach," Comput. Oper. Res., **36**(6), pp. 2082–2096.
[9] Wang, F.-S., and Chen, L.-H., 2013, "Heuristic Optimization," *Encyclopedia of Systems Biology*, W. Dubitzky, O. Wolkenhauer, K. Cho, and H. Yokota, eds., Springer, New York, pp. 885.
[10] Poudel, L., Zhou, W., and Sha, Z., 2021, "Resource-Constrained Scheduling for Multi-Robot Cooperative Three-Dimensional Printing," ASME J. Mech. Des., **143**(7), p. 072002.
[11] Yu, J., 2015, "Intractability of Optimal Multirobot Path Planning on Planar Graphs," IEEE Robot. Autom. Lett., **1**(1), pp. 33–40.
[12] Tereshchuk, V., Stewart, J., Bykov, N., Pedigo, S., Devasia, S., and Banerjee, A. G., 2019, "An Efficient Scheduling Algorithm for Multi-Robot Task Allocation in Assembling Aircraft Structures," IEEE Robot. Autom. Lett., **4**(4), pp. 3844–3851.
[13] Culbertson, P., Bandyopadhyay, S., and Schwager, M., 2019, "Multi-Robot Assembly Sequencing Via Discrete Optimization," 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, Nov. 3–8, IEEE,pp. 6502–6509.
[14] Cauligi, A., Culbertson, P., Stellato, B., Bertsimas, D., Schwager, M., and Pavone, M., 2020, "Learning Mixed–IInteger Convex Optimization Strategies for Robot Planning and Control," 2020 59th IEEE Conference on Decision and Control (CDC), Jeju, South Korea, Dec. 14–18, IEEE, pp. 1698–1705.
[15] Atay, N., and Bayazit, B., 2006, Mixed-Integer Linear Programming Solution to Multi-Robot Task Allocation Problem, Washington University, St. Louis, Tech. Rep, 2006.
[16] Darrah, M., Niland, W., and Stolarik, B., 2005, "Multiple UAV Dynamic Task Allocation Using Mixed Integer Linear Programming in a SEAD Mission," Infotech@Aerospace, Arlington, VA, Sept. 26–29, p. 7164.
[17] Wei, C., Ji, Z., and Cai, B., 2020, "Particle Swarm Optimization for Cooperative Multi-Robot Task Allocation: A Multi-Objective Approach," IEEE Robot. Autom. Lett., **5**(2), pp. 2530–2537.
[18] Sarkar, C., Paul, H. S., and Pal, A., 2018, "A Scalable Multi-Robot Task Allocation Algorithm," 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, May 21–25, IEEE, pp. 5022–5027.
[19] Zitouni, F., Maamri, R., and Harous, S., 2019, "FA–QABC–MRTA: A Solution for Solving the Multi-Robot Task Allocation Problem," Intell. Serv. Robot., **12**(4), pp. 407–418.
[20] Zhang, P.-Y., Lü, T.-S., and Song, L.-B., 2004, "Soccer Robot Path Planning Based on the Artificial Potential Field Approach With Simulated Annealing; Soccer Robots; Soccer Robots," Robotica, **22**(5), p. 563.
[21] Zhang, Q., Manier, H., and Manier, M.-A., 2012, "A Genetic Algorithm With Tabu Search Procedure for Flexible Job Shop Scheduling With Transportation Constraints and Bounded Processing Times," Comput. Oper. Res., **39**(7), pp. 1713–1723.
[22] Gendreau, M., Hertz, A., and Laporte, G., 1994, "A Tabu Search Heuristic for the Vehicle Routing Problem," Manage. Sci., **40**(10), pp. 1276–1290.
[23] Thabit, S., and Mohades, A., 2018, "Multi-Robot Path Planning Based on Multi-objective Particle Swarm Optimization," IEEE Access, **7**, pp. 2138–2147.
[24] Padmanabhan Panchu, K., Rajmohair, M., Sundar, R., and Baskarair, R., 2018, "Multi-objective Optimisation of Multi-Robot Task Allocation With Precedence Constraints," Defence Sci. J., **68**(2), pp. 175–182.
[25] Lorpunmanee, S., Sap, M. N., Abdullah, A. H., and Chompoo-inwai, C., 2007, "An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment," Int. J. Comput. Infor. Eng., **1**(5), pp. 1343–1350.
[26] Sánchez-Ante, G., Ramos, F., and Frausto, J., 2000, "Cooperative Simulated Annealing for Path Planning in Multi-Robot Systems BT-MICAI 2000: Advances in Artificial Intelligence," Mexican International Conference on Artificial Intelligence, Cairó, O., Sucar, L. E., and Cantu, F. J., eds., Springer, Berlin, pp. 148–157.
[27] Balan, K., and Luo, C., 2018, "Optimal Trajectory Planning for Multiple Waypoint Path Planning Using Tabu Search," 2018 9th IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2018, IEEE, pp. 497–501.
[28] Jaillet, L., and Porta, J. M., 2017, "Path Planning With Loop Closure Constraints Using an Atlas-Based RRT," *Robotics Research*, H. Christensen, and O. Khatib, eds., Springer Tracts in Advanced Robotics, Vol. 100, Springer, pp. 345–362.
[29] Lejeune, E., Sarkar, S., and Jezequel, L., 2021, A Survey of the Multi-Agent Pathfinding Problem, Technical Report, Accessed July 2021.
[30] Werfel, J., and Nagpal, R., 2008, "Three-Dimensional Construction With Mobile Robots and Modular Blocks," Int. J. Robot. Res., **27**(3–4), pp. 463–479.
[31] Werfel, J., Petersen, K., and Nagpal, R., 2014, "Designing Collective Behavior in a Termite-Inspired Robot Construction Team," Science, **343**(6172), pp. 754–758.
[32] Sartoretti, G., Wu, Y., Paivine, W., Kumar, T. K. S., Koenig, S., and Choset, H., 2019, "Distributed Reinforcement Learning for Multi-Robot Decentralized Collective Construction," Distributed Autonomous Robotic Systems, N. Correll, M. Schwager, and M. Otte, eds., Springer Proceedings in Advanced Robotics, vol. 9. Springer, Cham, pp. 35–49.
[33] Ortiz, C. L., Vincent, R., and Morisset, B., 2005, "Task Inference and Distributed Task Management in the Centibots Robotic System," Proceedings of the International Conference on Autonomous Agents, pp. 997–1004.
[34] Peres, J., Rosa, P. F. F., and Choren, R., 2018, "A Multi-Agent Architecture for Swarm Robotics Systems," Proceedings—2017 IEEE 5th International Symposium on Robotics and Intelligent Sensors, IRIS 2017, pp. 130–135.
[35] Badreldin, M., Hussein, A., and Khamis, A., 2013, "A Comparative Study Between Optimization and Market-Based Approaches to Multi-Robot Task Allocation," Adv. Artif. Intell., **2013**, pp. 1–11.
[36] Poudel, L., Zhou, W., and Sha, Z., 2020, "A Generative Approach for Scheduling Multi-Robot Cooperative Three-Dimensional Printing," ASME J. Comput. Infor. Sci. Eng., **20**(6), p. 061011.
[37] Werfel, J., Petersen, K., and Nagpal, R., 2011, "Distributed Multi-Robot Algorithms for the TERMES 3D Collective Construction System," Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011), pp. 1–6.
[38] Elagandula, S., Poudel, L., Sha, Z., and Zhou, W., 2020, "Multi-Robot Path Planning for Cooperative 3D Printing," Proceedings of the ASME 2020 15th International Manufacturing Science and Engineering Conference, p. V001T01A034.
[39] Poudel, L., Marques, L. G., Williams, R. A., Hyden, Z., Guerra, P., Fowler, O. L., Moquin, S. J., Sha, Z., and Zhou, W., 2022, "Toward Swarm Manufacturing: Architecting a Cooperative 3D Printing System," ASME J. Manuf. Sci. Eng., **144**(8), p. 081004.