Traveling Cellsman: Partition-Cluster Co-Parameterization for Multi-robot Cooperative 3D printing

Matthew Ebert[a], Ronnie Stone[b], Ergun Akleman[c,d], Zhenghui Sha[b], Vinayak Krishnamurthy[a,d,*]

[a]*J. Mike Walker '66 Department of Mechanical Engineering, Texas A&M University,*
[b]*Walker Department of Mechanical Engineering, University of Texas at Austin,*
[c]*Department of Visualization, Texas A&M University,*
[d]*Department of Computer Science and Engineering (by affiliation), Texas A&M University,*

Abstract

We present Traveling Cellsman, an approach for creating a parameterization for task scheduling and collision avoidance with Cooperative 3D printing (C3DP). The parameterization is based on the distribution of work between robots (partition), which allows the robots to navigate through their printing tasks effectively while also allowing for collision avoidance with other robots. The parameterization provides straightforward optimization of makespan. Inspired by the multiple traveling salesman problem (MTSP), we schedule tasks by first clustering tasks together based on a parameterization of the partition. The clustered tasks can then be ordered for printing. Numerical results indicate that our clustering approach finds an optimal solution faster than the non-clustered approach for minimizing the pause and movement time of the robots. Physical results also show that optimization allows for faster printing time as compared to non-optimized or slicer-based methods for generating a printing schedule. While we demonstrate our method using C3DP, it is generally applicable to other multi-robot task scheduling problems where collision may occur.

1. Introduction

Multi-robot cooperative 3D printing (C3DP) is a growing area of research. The motivation behind C3DP is to achieve efficiency, flexibility, and scalability in additive manufacturing by leveraging simultaneous deposition of material at multiple locations for a given part (Poudel et al., 2020a). Despite the promise of C3DP, the so-called cooperation between different robots is inherently challenging due to several interlinked and often conflicting objectives such as collision avoidance, task scheduling, part segmentation, and so on. Accordingly, there have been several investigations of each of these problems individually or in combination with each other.

A critical gap in current research on C3DP is the lack of a principled approach that encapsulates the myriad challenges within a common formal framework. To address this gap, we present a topologically-guided framework for spatial reasoning in C3DP. Our framework focuses on layer-wise cooperation across an arbitrary number of robots tasked with printing arbitrarily shaped parts. The core idea behind this framework is the notion of *spatial parametrization induced by the partitions of a given layer* assigned to each robotic printer. When applied to layers with internal cellular structures (as is typically the case with infills), this parametrization leads to an elegant formulation of the makespan (total time to print) optimization problem as a novel and previously unknown variation of the traveling salesman problem, thereby motivating the name "*Traveling Cellsman*" (Fig. 1).

*Corresponding author
Email address: vinayak@tamu.edu (Vinayak Krishnamurthy)

1.1. Technical Problem & Scope

From a broad perspective, one can identify two key algorithmic components in any C3DP workflow. The first is division of work, i.e., the decomposition of the volume of a part into sub-volumes each of which, is assigned to a printing robot. There are many different methods for creating sub-volumes, such as creating angled sub-volumes (Poudel et al., 2018; McPherson and Zhou, 2018; Weber et al., 2022), or creating interlocking sub-volumes (Kuipers et al., 2022; Ebert et al., 2025; Krishnamurthy et al., 2022). The second algorithmic component is scheduling, which may involve sequencing of cells within a layer, determining the path of printing nozzles with precise labeling of which parts of the nozzle path correspond to material extrusion, movement without extrusion, and idle time (e.g. while pausing). The objectives, for each of these tasks, could vary ranging from maximizing cooperation between printers to the minimization of makespan under essential constraints such as collision avoidance. In many cases, the segmentation itself imposes restrictions or constraints on the printing schedule such as task dependency of sub-volumes (Poudel et al., 2020b). Other layer-wise collaborative efforts create "safezone-style" regions where only one robot may print at a time (Stone et al., 2024, 2025; Krishnamurthy et al., 2022).

We observe that the common element that connects these issues is that of determining the right spatial parametrization that simultaneously captures division of labor and scheduling and enables a good formalization of desirable objectives (e.g., makespan) under given constraints (e.g., collision avoidance). In our work, we specifically consider the scenario where multiple stationary robots cooperatively print in a layer-wise manner,

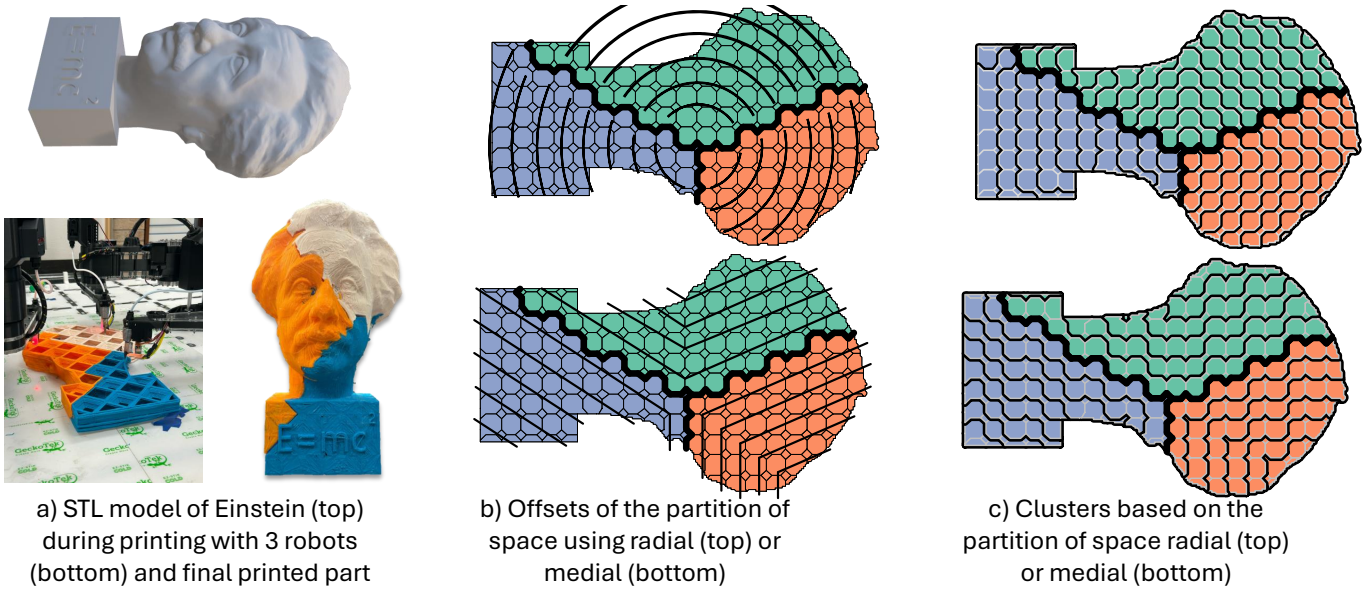


Figure 1: The Traveling Cellsman algorithm enables cooperation between robots through parametrically clustering the design space and allowing for optimization of a robots schedule.

and we assume a global-local control and coordination that enable robots to pause upon the detection of proximity with other robots. With these assumptions, our technical problem is to determine such a parametrization and also the methodology to subsequently use the parametrization to minimize the makespan (objective) while avoiding collision (constraints).

1.2. Approach and Rationale

The main focus of this work is to develop a methodology to determine the sequence in which the cells in a given layer are printed. The design space of sequences, as such, is combinatorially large. We observe that our problem shares a remarkable resemblance with a variation of the TSP called the multiple Traveling Salesman Problem (MTSP) (Lawler, 1985; Bektas, 2006) (Fig. 2 a)). Unlike standard MTSP, where tasks can be freely reassigned to salesmen (robots in our case), our problem is more closely related to a colored MTSP approach where cells are preassigned to robots due to kinematic and positioning constraints (Li et al., 2014b,a; Pandiri and Singh, 2018). For our problem, the coloring of cells is dependent on the partition of the printing space, only robots can print cells within their partition (Fig. 2 b)). To manage the tradeoff between combinatorial complexity and objective performance, additional works have further superimposed the notion of clustering that, in our case, would correspond to grouping cells in order to reduce the size of the design space when sequencing the tasks (Fig. 2 c)) (Nallusamy et al., 2010; Bao et al., 2023). Our approach in this work stems from the question: "**How should we cluster cells so as to determine the optimal sequence in which cells should be printed without compromising on the objective performance?**"

We begin by observing that the interfacing boundaries between sub-volumes assigned to their respective printers naturally lend themselves to a parametrization that uniquely char-

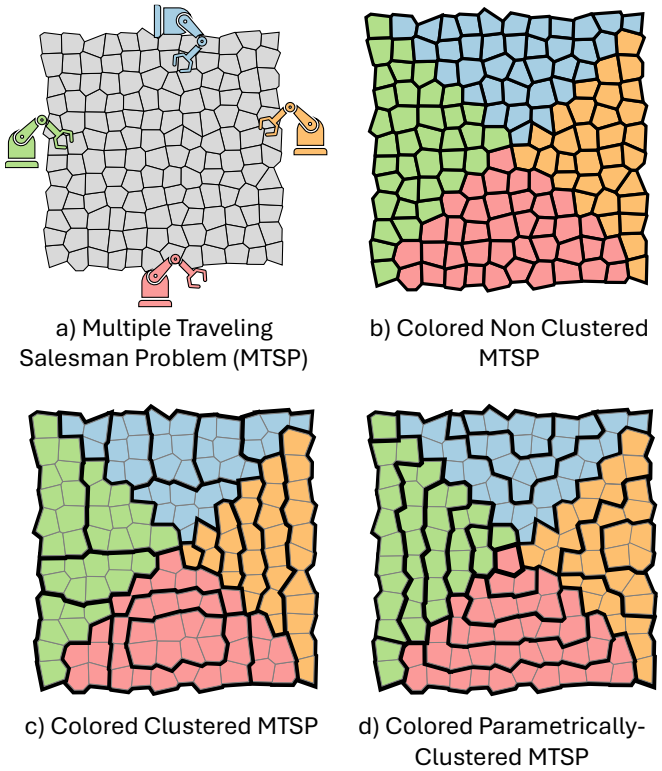


Figure 2: Different variations of a TSP problem where tasks are not preassigned a), preassigned to specific robots (colored) b) (Li et al., 2014b,a), clustered into groups and preassigned to specific robots (colored-clustered) c) (Nallusamy et al., 2010; Bao et al., 2023), or clustered using a spatial parameterization (colored-parametrically clustered) d).

acterizes distances from those boundaries. Formally, the level-sets of the distance field from the boundaries offer a direct way to determine the gross movement of each printing nozzle. Note that for layers subdivided into cells, this parametrization essentially translates to task scheduling by virtue of discretization. With this in view, **our broader approach develops an interface-induced parametrization which can then be used as a representation to compute optimal sequence of cells.** The crux of our approach is to use this parametrization as the means to generate clusters for colored clustered MTSP. We call this approach **parametric clustering** (Fig. 2 d)).

1.3. Key Contributions

The first and fundamental contribution of this work is the development of *interface-induced parametrization and clustering methodology* for scheduling of tasks in layer-wise cooperative 3D printing (C3DP). This theoretically grounded methodology offers a natural representation for task scheduling for minimization of total printing time with C3DP. To this effect, our second contribution is the use of parametric clustering for the development of a novel computational framework for minimizing the total time to print. This framework, inspired by clustered multi-color traveling salesman problem is implemented using genetic algorithms (GA). The key novelty of our framework is that it elegantly leverages our idea of parametric clustering to encode the printing sequence of cells in terms of two types of clusters, namely, radial and medial clusters of cells. Furthermore, we make a secondary contribution by developing a simplified area-based function for evaluating the print time to offer significant computational efficiency during GA-based optimization.

In terms of insights, our comprehensive numerical comparisons show that our parametric clustering arrives at a better optimal schedule much faster than a non-clustered sequencing approach in GA-based optimization. We also show that by changing the clustering type from medial to radial (or vise-versa), the time to print can be reduced. Through physical evaluation with three robots we found that an optimized strategy allows for more parallel printing than a non-optimized and a slicer-based approach. For example, for a long part (approximately 850*mm* x 270*mm*) cooperatively printed using 3 robots, our method helps save 8.5 minutes on average per layer leading to hours of time saved across the entire volume.

2. Background

2.1. Collaborative Robot Systems

There has been extensive prior research on systems which have multiple robots which must collaborate in some way. The amount of collaboration between robots varies based on the system While it is also common to Some areas where collaborative systems are used are coverage tasks (Lin and Huang, 2021; Gul et al., 2022), assembly/disassembly in industrial robotics (Keshvarparast et al., 2024; Marvel et al., 2018; Tereshchuk et al., 2019), and painting/spraying tasks (Zbiss et al., 2022, 2024).

While this paper is generally focused on multi-robot systems, we use collaborative 3D printing(C3DP) to validate where multiple robots must work together to share tasks and avoid collision. There are various different C3DP application systems with different robot geometries and setups with some using gantry systems, moving robots, robotic arms, welding wire arc robots, or flying robots (Alhijaily et al., 2023b; Dörfler et al., 2022; Sustarevas et al., 2019; Zhang et al., 2022, 2018; Kuipers et al., 2022; Arbogast et al., 2024; Bhatt et al., 2022). In these cases robots print in either planer (Poudel et al., 2020a) or non-planar (Bhatt et al., 2019) manner with multiple robots with various degrees of freedom. Robots printing tasks are within close proximity since they are working on a common part which can make collision avoidance a difficult challenge.

2.2. Scheduling for Collaborative Systems

In many cases when robots work in close environments it is preferable for the robots to work as independently as possible (Larsen et al., 2015; Fei et al., 2004; Lin and Huang, 2021). These have been compared before to CPU systems where most/all tasks are parallelized without much difficulty (Zhang and Parker, 2013). Other methods exist for systems with temporal constriants (Gombolay et al., 2013), specific manufacturing constraints (Liu et al., 2023b), or with cluttered environments (Xidias et al., 2010).

Within C3DP systems tasks have been scheduled using various methods such as scheduling paths by creating subpaths (Jin et al., 2019; Alhijaily et al., 2023b,a), or by scheduling subvolumes (Poudel et al., 2021, 2020a; Elagandula et al., 2020). Other methods involve labeling specific regions as safe/unsafe and then sequentially printing the unsafe parts followed by collaborative printing of safe regions (Krishnamurthy et al., 2022; Ebert et al., 2025), others try to schedule the printing of unsafe and safe regions without pausing (Stone et al., 2024, 2025). These are either limited by having large sub-regions which can be difficult to schedule well, or a low amount of collaborative printing.

Scheduling is not required for multi-robot systems, instead ordering of work can be made independently and collision can be avoided by constantly checking robot position. This can lead to high overhead to collision avoidance and non-optimal printing (Stone et al., 2023).

2.3. Traveling Salesman for Task Scheduling

The traveling salesman problem (TSP) has been a popular problem in computer science (Lawler, 1985) with applications in robot task scheduling, where tasks are treated as points and distance is minimized. With variations such as multiple TSP (MTSP) (Bektas, 2006), where some have applied it to multi-robot task allocation and scheduling problems for optimization (Fig. 2 a)) (Zacharia and Aspragathos, 2005; Kitjacharoenchai et al., 2019; Liu et al., 2023a).

However, in many MTSP problems with multiple robots, collision is not a consideration, and most problems can be reduced to a series of TSPs solved separately (Zhang and Parker, 2013; Nallusamy et al., 2010). While this may be useful, it is often not the case that robots can work separately since they may

share workspaces. One variation of the MTSP called the colored MTSP aims to reduce the possible solution space by assigning some, if not all, cities to certain salesmen (Fig. 2 b)) (Li et al., 2014a,b; Meng et al., 2017; Nallusamy et al., 2010). This is particularly applicable to many robotic task allocation jobs where only some robots can a mplish a task, due to manipulator requirements, tool availability, or reachability of the robots (Li et al., 2014a; Wei et al., 2020).

Since TSP and MTSP are combinatorial problems, various methods exists to obtain optimal or near-optimal solutions. Most approaches uses a metaheuristic optimization strategy such as simulated annealing, particle swarm optimization, or genetic algorithms (Kong et al., 2019; Carter and Ragsdale, 2006; Zacharia and Aspragathos, 2005; Nallusamy et al., 2010). In many cases, the design of the chromosome for genetic algorithms allows for the swapping of tasks between robots without increasing the design space(Carter and Ragsdale, 2006; Király and Abonyi, 2010, 2011; Meng et al., 2017).

Another method for solving TSP is to cluster tasks into groups and then order the clusters, since the number of clusters is less than the number of tasks, the design space is reduced (Fig. 2 c)) (Chandran et al., 2006; Liao and Liu, 2018; Martin et al., 2023). A common method for clustering is to use k-means clustering, which creates groups of points that are close to one another (Alitappeh and Jeddisaravi, 2022). Another use of k-means clustering is to assign tasks in MTSP where the number of clusters matches the number of robots. After the assignment, the order for each robot is optimized independently (Nallusamy et al., 2010). While this is effective in grouping tasks, it groups tasks that are close to each other, which may not allow for collision avoidance.

There has also been work on solving TSP problems for 3D printing. One work uses TSP to order printing paths to minimize the number of retractions during printing which can take a long time, especially in DED or concrete printing (Singh et al., 2022; Sahu and Panda, 2024). These works place points within the object being printed and then use these points to traverse and optimize printing while minimizing the number of retractions needed. Other works use given paths and optimize the traversal between paths (Ganganath et al., 2016).

While the TSP and other scheduling methods have been applied before to collaborative robot systems, there currently exists a gap in literature for when there are a high number of tasks that must be scheduled without collision. When the number of tasks is reasonable, the number of combinations is reasonable to solve. In the case of 20 tasks, there are 2.43×10^{18} combinations for a single robot or 7.25×10^{6} combinations for two robots with equal distribution of tasks. However, if there are 100 tasks, then the problem is likely intractable. With single robot TSP, there are 9.33×10^{157} combinations, and with two robot preassigned MTSP, there are 6.08×10^{64} combinations. This is the gap that our work fills by parametrically clustering space using the coloring of the tasks so that an optimal solution can be found quickly (Fig. 2 d)). The closest terminology within the TSP space for this problem would be a Colored Parametrically-Clustered MTSP.

2.3.1. Complexity Analysis

General TSP problems have a solution space of $N!$ where N is the number of tasks (Meng et al., 2017; Nallusamy et al., 2010). This can be reduced if the space is symmetric, meaning that traversal from point 1 to point 2 is identical to traversal from point 2 to point 1. This is not the case when a collision must be considered. Therefore, the solution space cannot be reduced (Nallusamy et al., 2010). For MTSP, where tasks are not pre-assigned, then the number of combinations is still at least $N!$ depending on chromosome representation (Király and Abonyi, 2011; Li et al., 2014b). If tasks are preassigned to m robots then the solution space is $\prod_{k=1}^{m} |N_k|!$, where $|N_k|$ is the number of tasks assigned to the k^{th} robot (Meng et al., 2017). As mentioned before one method to reduce the solution space is to cluster tasks which reduces N_k if there are more than 1 task per cluster. This can be very useful when there are a high number of tasks. As an example, if there are 300 tasks, the solution space for TSP is 3.06×10^{614} and with 3 robots and equal distribution of tasks the solution space becomes 8.128×10^{473}. If each robot has 10 clusters, then the solution space becomes a more reasonable 4.778×10^{19} possible combinations.

3. Conceptual Framework

Our conceptual framework starts with a few assumptions about the robot-task space. The first assumption is that all tasks are located on surfaces (2-manifolds). This assumption is consistent with a layer-wise printing approach, usually seen in additive manufacturing applications. While we only show results for flat horizontal surfaces, it is possible for the same algorithm to work on curved orientable surfaces. The second assumption is that the tasks are located within some finite region. The robots are assumed to be connected with a home position fixed in space with kinematic constraints, in our case a two-linkage mechanism connects the home, and nozzle position.

We first define the mathematical framework for parametric clustering, which uses level sets of the partition to form medial or radial clustering. Then we discretize since our partitioning consists of cells to be printed and use the same approach in order to generate radial or medial clusters of space.

3.1. Interface induced Parameterization

To segment the workspace and assign tasks to robots we use a Voronoi decomposition, where each task is assigned to the closest robot. While simplistic, this is a natural segmentation method to ensure reachability if the robots are stationary or have limited movement capabilities from their home positions. We can define the partition notation of P_i which is the set of tasks assigned to robot i so that $P_i \subset \Omega$ is a subset of all tasks Ω where $\bigcup_i P_i = \Omega$ and $P_i \cap P_j = \emptyset \quad \forall i \neq j$.

The Voronoi decomposition also allows for the formation of edges ε and vertices ν, where edges are at equal distances from any two robots, and vertices are at equal distances from three or more robots. We name this Voronoi decomposition as the partition of the printing space, since it defines task allocation using only vertices and edges for any number of robots.

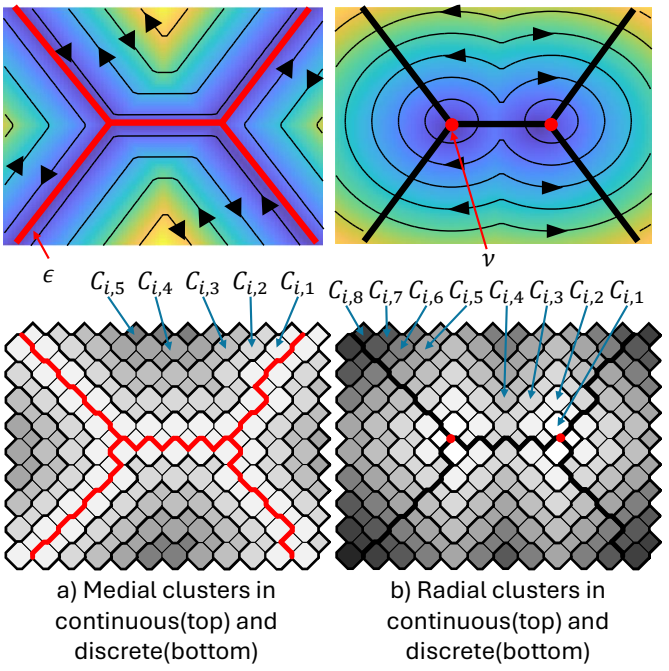


Figure 3: The space can be parameterized using the partition a) either radial b) or medial c) based on the distance from the partition vertices or edges respectively. This is the same in the discrete space with cells (bottom).

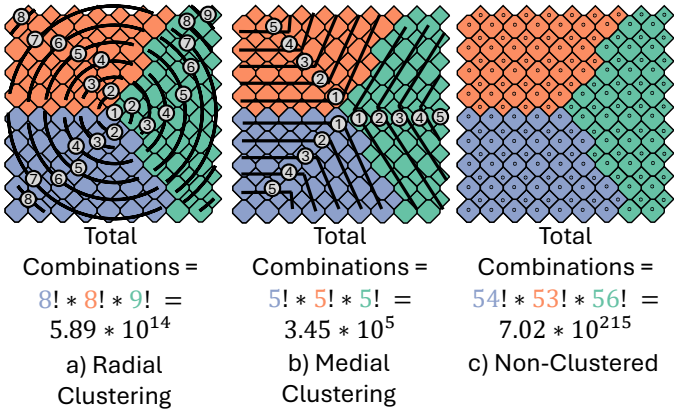


Figure 4: The total number of combinations for solving the TSP using Medial, Radial, or no clustering.

3.2. Parameterization

Following the partitioning of the printing space and the subsequent task allocation, we proceed to the parameterization process. First, we assume that we are operating in a continuous space, where there are an infinite number of tasks within a defined region. Using two distance functions, we can cluster tasks using medial or radial parameterization based on the partition geometry.

3.2.1. Medial

The first parameterization space (medial) uses the p function where $p(x) = min_{e \in \varepsilon} \|x - e\|$ gives the distance from a point x to the closest edge ε. The clustering can then be defined on level sets of the equation where all points of equal p are within the same cluster. Since the partition defines the assignment of tasks, $C_{i,k} = \{p(x) = k, x \in P_i\}$. Using this parameterization $C_{i,k}$ results in offsetting bands from the edges of Voronoi decomposition (Fig. 3 a)).

3.2.2. Radial

The second parameterization of space (radial) organizes clusters based on the distance function $q(x) = min_{v \in \nu} \|x - v\|$ where ν is the vertices of the partition. This creates clusters at the level sets of the function $q(x)$ represented by $C_{i,k} = \{q(x) = k, x \in P_i\}$ This forms clusters that are concentric regions around the vertices of the partition (Fig. 3 b). The vertices ν can also be described as the branching points of the medial edges, since they are the vertices of ϵ where 3 or more edges meet.

3.3. Segmentation into Cells

While the proposed parameterization method applies to an infinite number of tasks, real robotic task allocation work involves a finite number of tasks in a compact domain. In this case, the time to complete a printing task, the traversal time between tasks, and the robot position at any point in time are known. To create a printing sequence, we first start with the assumption that all printing tasks are organized into a cellular tessellation similar to prior work on Chunking, LayerLock, and NoodlePrint. Formally, the tessellation can be defined as a set of cells/faces F and vertices V, where P_i represents all the cells (tasks) assigned to robot i.

To formulate the parameterization for this tessellation, the partition edges ϵ and ν must be defined. The edges for a given cell are denoted by $E(f)$ for cell f. The partition edges are defined by $\epsilon = \{e \in E(f_1) | f_1 \in P_i, f_2 \in P_j, i \neq j, e \in E(f_2)\}$, meaning ϵ is formed by edges shared by cells assigned to different robots. Using terminology from prior works, the ϵ defined here is called the interfacing complex/path in prior works. Thus, the term ϵ and interfacing complex/path will be used interchangeably. (Krishnamurthy et al., 2022; Ebert et al., 2025).

To form the vertices ν in a cellularized tessellation, let us first denote the vertices by $V(f)$ for any cell f. The interfacing vertices ν are then defined as $\nu = \{v \in V(f) | v \in V(f_1) \cap V(f_2) \cap V(f_3), f_1 \in P_i, f_2 \in P_j, f_3 \in P_k, i \neq j \neq k\}$. The vertices ν can also be described as the branching points of the interfacing complex ϵ.

While the previous continuous space formulation relies on distance to partition edges, the partition edges for the cellular case may not be smooth/straight and also may not have cells at a specific distance. For this reason, cluster assignment is defined by topological connectivity to the partition edges/vertices, which is the minimum number of adjacent faces which must be traversed to reach an edge/vertex.

3.3.1. Medial Clustering

The first cluster using medial parameterization is formed by the cells which neighbor the partition edges which can be defined as $C_{i,1} = \{f \in P_i \mid E(f) \cap \epsilon \neq \emptyset\}$. The following

clusters are defined based on cell connectivity to ϵ, meaning $C_{i,k} = \{f \in P_i \mid f \notin C_{i,k-1}, E(f) \cap E(C_{i,k-1}) \neq \emptyset\}, \quad k \geq 2$. This means that the cells in cluster $C_{i,k}$ neighbor $C_{i,k-1}$ and do not belong to $C_{i,k-1}$ or any other lower cluster number (Fig. 3 a)).

3.3.2. Radial Clustering

In a similar topological manner, the radial clusters can be determined based on if they neighbor the vertices v and then build later clusters based on topological connectivity to prior clusters. The first cluster is defined by $C_{i,1} = \{c \in P_i \mid V(c) \cap v \neq \emptyset\}$, which are all cells that neighbor the branching points of the interfacing complex.

The following clusters are then defined by $C_{i,k} = \{c \in P_i \mid V(c) \cap V(C_{i,k-1}) \neq \emptyset,\ c \notin C_{i,k-1}\}, \quad k \geq 2$ (Fig. 3 b)).

4. Algorithm

4.1. Parameterized Cell Sequencing

While the formation of the clusters through medial or radial parameterization allows for a smaller design space, that is, fewer possible combinations when ordering the tasks, the clustering does not inherently avoid collisions. Although collision can be avoided in real-time, a poor ordering of the clusters can result in long print times, as the number of pauses increases. Thus, we optimize the overall print time by using a genetic algorithm (GA) to select the ordering to visit the clusters. This is similar to many other TSP problems where the sequencing of cities/tasks can be modified depending on their impact on the improvement of distance/time.

To represent the ordering of clusters within the GA framework, we start by defining the clusters for all robots, either in a medial or radial fashion. The number of clusters for each robot is labeled as G_i for robot i which is a positive integer. The clusters for each robot can be labeled according to their cluster number c, their respective robot i, and the order in which they will be printed p, by adopting the notation $C^p_{i,c}$ (Fig. 5). As an example, $C^5_{1,3}$ indicates that cluster 3 of robot 1 is the 5th cluster to be printed. Using this notation, the schedule for a robot can be represented by $S_i = [C^1_{i,1}, C^2_{i,2}, ...]$ which denotes the initial print order, i.e. cluster one is printed first, cluster two second, and so on (Fig. 5 b) and c)). Furthermore, the entire schedule S can be represented by $S = [S_1, S_2, ..., S_m]$ (Fig. 5 c)) if there are m robots with a solution size of $\prod_{n=1}^{m} S_n!$. As an example, if there are three robots with 54, 53, and 56 tasks each, the total number of combinations for a non-clustered TSP is 7.02×10^{215} (Fig. 4 c)). Using radial or medial clustering reduces the design space to 5.89×10^{14} and 3.46×10^5 respectively (Fig. 4 a) and b)). This chromosome representation is very similar to the setup described by Carter et al., which is described as a "two-part chromosome" Carter and Ragsdale (2006) or the "Path-city number" chromosome described by Jiang et al. (Jiang et al., 2020).

While the order of printing clusters may change, it is important to follow the rule that no two clusters for a robot can have the same schedule number, that is, $C^j_{i,c} \neq C^j_{i,c'} \forall c \neq c'$. It follows that the scheduling order for all clusters must be integers between 1 and G_i (Fig. 6).

4.1.1. Crossover

Typical crossovers in the GA framework involve selecting two parent chromosomes and swapping parts of their genes. This would not work in our case, since it would not ensure that each cluster is selected only once. Instead, in our case, crossovers affect only the schedule S_i of a single robot. We first select a random integer between 1 and $\lfloor (G_i/2) \rfloor$, then select two non-overlapping portions of the S_i with the size of the random integer. These two non-overlapping portions of S_i can then be swapped. Finally, each switched portion can also be randomly flipped to form the chromosome (Fig. 7).

This crossover methodology maintains that all numbers between 1 and G_i exist within S_n, meaning that all clusters are visited exactly once, while also allowing for different schedules. This is similar to only allowing for the "in-route mutation/crossover" described in prior works (Király and Abonyi, 2011).

While another common method for modifying a chromosome within the GA is a mutation, in our problem formulation this is equivalent to only swapping two indices of schedule S_i since a single index cannot be modified alone without violating the property of all clusters being visited exactly once (Fig. 7 a) S_3). From our initial experiments, including a separate mutation function did not show a significant difference in results.

4.2. Collision Avoidance

Through changing the print schedule, it is possible to avoid most collisions, but it is also likely that there are no solutions that completely avoid collision due to part geometry, robot locations, or robot kinematics. For example, if a small part is being printed by many robots, the robots will all be working around the same region, which can lead to unsafe printing conditions. Because of these cases, the robots are always in communication with a central hub, which can determine if two robots are about to collide and send a pause command to one of the robots. This is similar to prior work in the area (Stone et al., 2023). While there is an upper limit on the number of times robot locations may be checked (frequency) based on hardware, we assume for this work that the current location of robots may be checked 10 times a second and once told to pause to avoid collisions, robots stop their movement and pause within 10ms. Once a robot receives a pause command, it must wait for 30 seconds before trying to continue. The safe distance for collision is 3 times the nozzle radius and the priority regarding which robot should pause in a two-robot collision situation is determined a priori.

4.2.1. Mixed Parameterization

In the description of the chromosome, the clustering methodology was chosen a priori. However, it is possible in some scenarios that it is not possible to reasonably intuit which may be preferable (medial or radial), especially with a high number of robots. Thus, we modify the chromosome to allow for changing the type of clustering within the chromosome.

We start by pre-computing the radial and medial clusters and storing them. For each robot, we then determine G_i for each robot by checking how many medial and radial clusters there

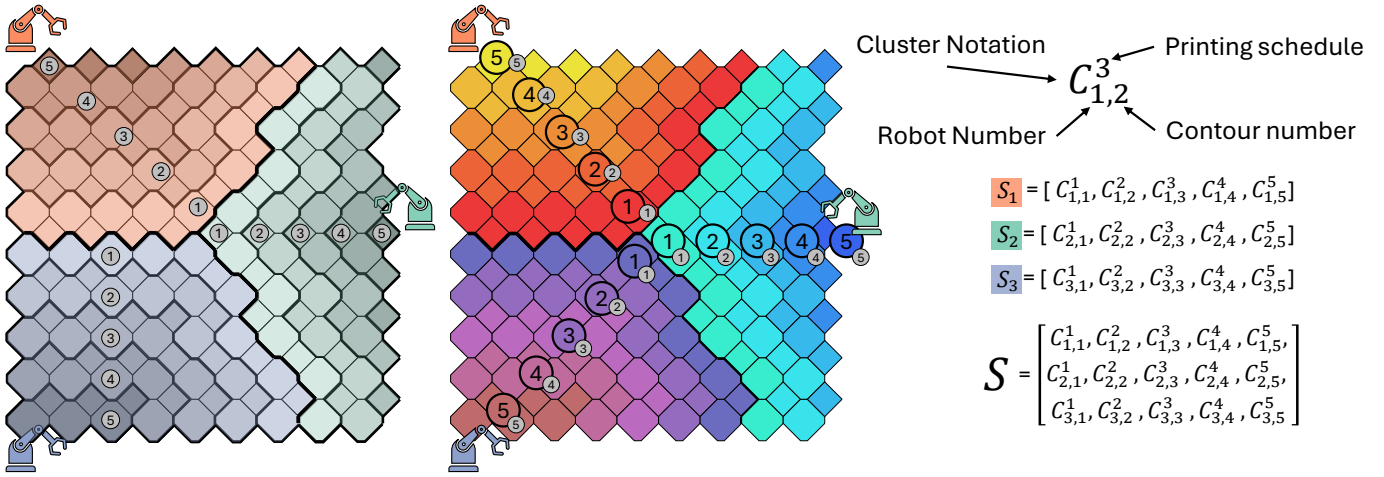


Figure 5: In order to represent the order in which clusters of tasks are scheduled, we first number all clusters (a) and then create a schedule of when clusters will be visited (b). Using this ordering per robot we can create a chromosome that describes the other in which clusters will be scheduled.

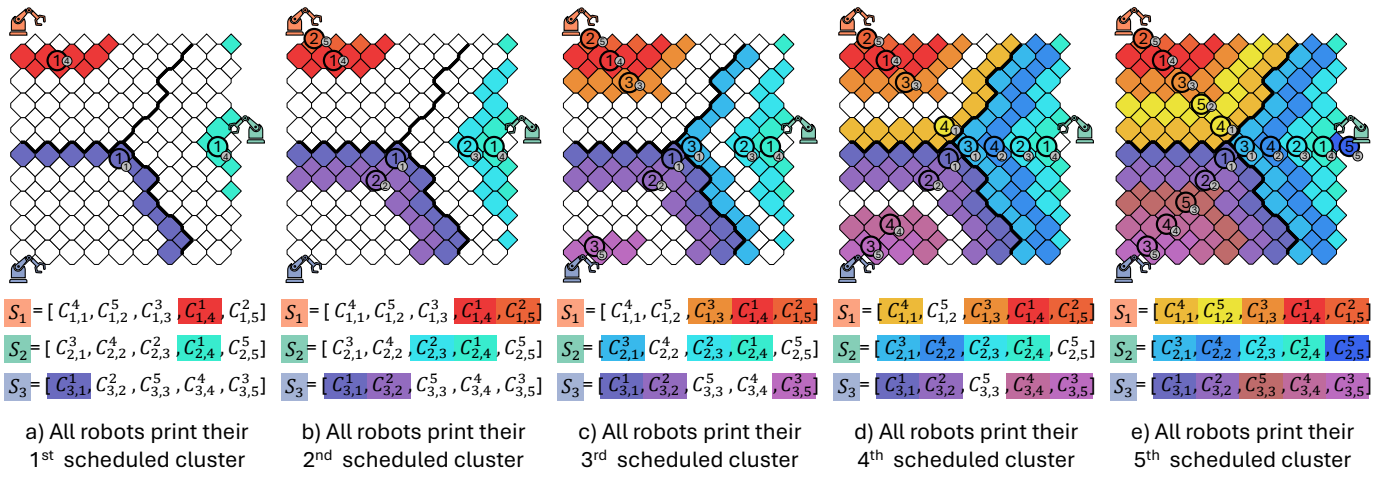


Figure 6: The order which robots print their clusters. Some clusters may take longer to print than others depending on the amount of printing required, this is an illustrative example.

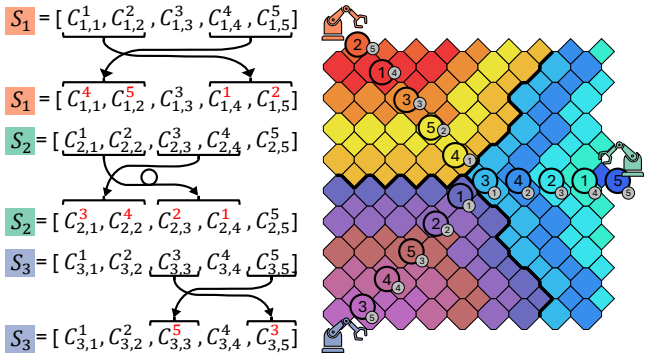


Figure 7: How crossover happens within different schedules by either direct swapping (S_1), swapping as well as reversing ordering (S_2), or swapping the schedule order of just two clusters (S_3).

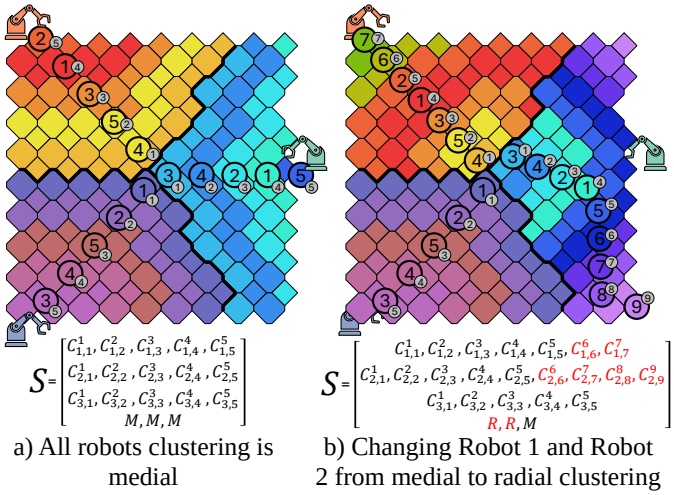


Figure 8: A representation of how changes in the clustering method can occur within a chromosome and how the change affects sequencing. The original chromosome is taken from Fig. 7.

are and taking the maximum value between them. The chromosome is then represented by an additional m values which represent medial or radial selection ("M"= medial, "R" = radial). As an example, a chromosome for 3 robots can be represented by $S = [S_1, S_2, S_3, R, R, M]$ which means that the clusters of the first two robots are radially parameterized, and the clusters of the last robot are medially arranged (Fig. 8). Some clusters may contain no cells if there are unequal numbers of radial and medial clusters. As an example, if there are 5 medial clusters, but 6 radial clusters for robot 1 then $C_{1,6}$ contains no cells when medial, but contains tasks for radial clustering (Fig. 8 b).

In this chromosome representation, crossovers can include flipping the last m values of S to change the clustering method. From our initial testing, it was determined that making this change was rarely preferable. This is likely due to needing several generations to optimize a given clustering method and changing often does not allow this to occur.

4.3. Printing Time Evaluation

In order to assess the chromosome choice, that is, the clustering and its associated ordering, we have to compute the time it would take to print. While most TSP problems only require distance to compute the optimum, robot tasks must also take into consideration collision avoidance strategies and division of work. For this reason, we simulate the time to print a given sequence using two different methods that replicate the physical setup to different degrees of accuracy and time computation, namely the path-based and the area-based evaluations.

4.3.1. Path-Based Evaluation

The path-based evaluation relies on using the sliced path to compute the total time to print, which can be cumbersome. The path for each robot is computed based on the cluster scheduling to determine the cell printing order. The path for each cell can be computed by simply computing offsets of the cell to be printed, either completely filled in or with a set number of layers. Between the printing of each cell, the robot performs movements. Once the path for all robots is computed, each robot has its base position and the home position where it can return without collision. The printing process then begins with the robot moving from its safe position to the starting point of its path. Once that point is reached, we assume that it moves along the printing path with a set constant speed. At set intervals (similar to the robot communication setup), the robots check collision with other robots, which is limited to a frequency of 10 times per second to match the physical setup. The more often collision checks are made, the longer to compute since computing square roots frequently is computationally expensive. The most straightforward collision avoidance strategy is to check the distance between the end-effectors of any two robots, and assign a threshold value as a minimum acceptable distance. The choice of which robot is made to pause when a collision is detected is determined by a hierarchy (robot 1 pauses before robot 2 pauses etc.). More complicated collision-avoidance strategies can be used, such as arm-to-arm distance of robots to further verify collision avoidance. If a distance is below the given threshold, then a robot moves to the home position for a minimum of 30 seconds and returns to its path. Using this strategy, we compute the time a robot spends extruding material, moving between cells, pausing to avoid collision, and waiting to move to the next layer.

This method for print time evaluation can be costly to evaluate since distance checks for every robot, including arm-to-arm distance checks can be computationally expensive when evaluated multiple times a second.

4.3.2. Area-Based Evaluation

Contrary to path-based evaluation, an area-based time evaluation can also be computed since the printing time of a cell is known and the position of a robot while printing a cell is constrained within specific bounds (i.e., the face of the cell). This process starts by computing the polygon-to-polygon distance for all cells and taking the smallest value (i.e., the most conservative approach). The arm-to-arm distance can also be

computed for every cell based on the robot during the printing of the given cells using the boundary of the cell. The threshold for a pair of cells is then set to either the minimum cell distance or the arm-arm distance, whichever is smaller. The time to print a cell is computed based on the path that it is printed and the speed of the robot. Finally, the distance from every cell to every other cell can be computed which gives the time for a robot to move from one cell to any other cell given the speed of the robot.

The printing simulation can then begin by robots starting their assigned cells and checking for collision using precomputed cell values. As robots finish printing their cells, they move to the next cells using the precomputed values with no need to check for collision multiple times a second. The transition between clusters also consists of just transitions between cells. If the distance between two printed cells is lower than a threshold, then one robot moves to its safe position and waits for 30 seconds to continue. Each robot follows this process until all work is done while also keeping track of the time it spent printing, moving, pausing, and idling.

Note that most computationally expensive steps can be done a priori and then used during the optimization process to speed up convergence. Although this evaluation method is not as precise as the path-based method since the exact robot positions are not known, it allows for fast evaluation of printing time. A detailed comparison of the accuracy between the two methods is presented in later sections.

5. Numerical Studies

For the numerical results section, we use our GA framework to optimize the sequence of printing cells/clusters of cells. We utilize a GA since it is common for TSP problems, additionally, the optimization is highly sensitive since a single gene change may result in several minutes of pausing or none. Since a GA is a metaheuristic method, we use the following hyper-parameters (unless stated otherwise) for our set of numerical results, which were determined based on initial testing:

- Maximum number of generations - 5,000
- Population of a generation - 50
- Stall generations(amount of generations without change before convergences assumed) - 50

5.1. Fitness Function

The proposed chromosome representation for the printing schedule can be used directly in the GA framework and the subsequent evaluation of the printing time, either through the path- or area-based evaluators. However, the time to print is not a viable fitness function, as a large portion of the time spent in the process is due to the extrusion of the material, which cannot be minimized. We first start by tracking the how the robots spend their time during printing. There are 4 total activities that the robots can spend their time.

1. Extruding - The time robots spend completing their extruding/depositing material
2. Moving - The time a robot spends moving from one cell to another
3. Pausing - The time robots spend pausing to avoid collision with other robots, this includes time moving from its home position back to printing
4. Idling - The time a robot spends after all extruding tasks have been completed waiting for the other robots to finish so the next layer can be started

Since the extruding time is only determined by the geometry of the printing part, it cannot be minimized, and therefore is not part of the fitness function. Furthermore, the time a robot spends idling is a function of the distribution of work between robots (equal distribution of extrusion time) and the pause/move time. As an example, if one robot has high pause/move times, other robots may have high idle time since the robots have finished their extrusion tasks. Since this work assumes near equal distribution of work, the idle time is only dependent on the pausing and moving of robots and, therefore, not included in the fitness function.

This leaves only the moving and pausing time as part of the optimization function. This makes sense as the moving time is typically the only objective for a TSP problem, while the pausing time is typical in a multi-robot motion planning optimization problem. However, calculating the moving and pause time is not trivial since it is different per robot. There are a few options for formulating the fitness function with movement and pause time by taking the maximum or the average value of all robots' move/pause times. We only evaluate maximum and average since we want to optimize the values towards zero with all robots having equal weight of punishment for poor sequences. To test this, we formulate 4 variations of the objective function to determine the best function.

In general, it was found that for printing a circle geometry with various numbers of robots, it is preferable to use the average values of move and pause time. Since the optimization method is heuristic-driven, it may not be possible to know exactly why one formulation is better than another. Using the maximum or average values tends to allow for fast convergence on a solution, and they both converge to similar values, although they may be different sequences. The main reason why we believe the average is the best is because it will take into account all robot's pause/move time as opposed to using the maximum value may cause dramatic shifts in magnitude, making it difficult to converge.

For the remainder of this work, anytime a fitness function is used to calculate an optimum, the average move and average pause time will be used for evaluation.

5.2. Area vs Path Time Evaluation

In section 4.3 two different methods were discussed to evaluate the printing time, namely, area-based and path-based. The path-based method more closely imitates the real robot setup which involves traversal of the printing path to calculate collision but is therefore much more time intensive to calculate the

	3 Robots - Medial Clustering		3 Robots - Radial Clustering	
Fitness function	Number of Generations	Time to print	Number of Generations	Time to print
Max(Move) + Max(Pause)	103	17.323	159	17.738
Max(Move) + Avg(Pause)	60	18.046	171	16.987
Avg(Move) + Max(Pause)	72	18.159	143	17.602
Avg(Move) + Avg(Pause)	57	17.387	132	17.176

Table 1: Different fitness functions printing a circle with 3 robots using Medial clustering (robot setup shown in Fig. 9 right).

time to print. On the other hand, the area-based method makes assumptions about robot printing time, collision avoidance, and relies on pre-computing for avoiding collisions (cell-to-cell distance rather than point-to-point). While the area-based method is inherently faster, it is important to determine whether they both reach the same maximum and do not result in a different time to print, meaning the optimal of one should be the optimal of the other.

The first test consists of running 30 different optimizations (different initial sequences) using the area-based time method and stopping the optimization early (after only 50 generations) which is before convergence. We then compare the time to print, pause, and moving time using path and area-based methods. By comparing the difference we can observe if the area-based is a reasonable estimation of the path-based method. The hypothesis that will be testing is that the total time to print, pause time, and move time are statically similar and then when a linear regression is run will have a slope equal to or near 1. We compare two cases where we are printing a circle with 2 and 3 robots with various cell sizes.

The results show that, at least qualitatively, the time to print matches for 2 and 3 robots and similarly with the pause and print time, this can be seen since all points lie near the dotted line which is the ideal case if the two mirrored each other exactly (Fig. 9). One example is with two robots, all sequences resulted in zero pause time with both area based and path based which was exact matching of the 150 data points. Additionally it is seen that the move time generally increases for smaller cells because as the number of cells increases the traversal between cells must also increase, this is generally true for TSP problems if more cities are added.

While a qualitative comparison is good, linear regression can also be run on the data to determine the slope and the variance. With two robots and the total time to print the slope is 0.9539 and an $R^2 = .84$, the move time has a slope of 1.0528 and an $R^2 = 0.98$, since the pause time data all lies on the same point linear regression cannot be run but the data matches exactly. For three robots, the total print time has a slope of 1.001 and an $R^2 = 0.99$, the pause time has a slope of 0.1577 and an $R^2 = 0.09$, and the move time has a slope of 1.0174 and an $R^2 = 0.93$. From this we notice that the area based is not great at estimating the pause time and generally tends to overestimate the pause time for the robots. This is due to the assumptions made where comparing cell-cell distance is smaller than nozzle-nozzle distance. Both the move and total time to print for three robots shows very good agreement between the area and the path based. From the two and three robot case, we fail to reject the null hypothesis, meaning the data does not provide strong enough evidence to conclude that the times differ significantly between the two cases.

This test however does not provide evidence that if given the same starting sequence, both methods would converge on the same optimal sequence. While both methods do not converge on the same sequence we show that they converge to similar sequences. In order to compare two sequences we will use Kendall τ_a measurement which compares two sequences on a scale -1 to 1 where 1 is exactly the same and -1 is nothing similar. Running the same setup as before we compare difference in cell printing (inserting the actual cells for printing into the cluster ordering) with 2 robots printing, Kendall's $\tau_a = 0.6398$ for Robot 1 and $\tau_a = 0.1573$ for robot 2. While these are not close to 1 and therefore we cannot say they are similar, they are also not completely dissimilar. From further testing it appears as though the sequences deviate in the first generation, meaning in the initial generation of 5 chromosomes, different sequences are preferred due to slight differences and mutated to continue. While it cannot be shown that the two different methods arrive at the same optimal sequence, the optimal they arrive at appear to also be near the optimal of another.

For the remainder of this paper, any time the time to print is used to optimize, the area based method is used unless otherwise specified.

5.3. Effect of Parameterization

One of the main contributions of this paper is the parameterization of tasks for multi-robot printing. To demonstrate the importance of parameterization, we compare the two different clustering strategies presented in this work, namely medial and radial, to a non-clustered strategy which would be a traditional MTSP problem without clustering. This will also be compared to a combination approach where one robot may use radial parameterization, and the other may use medial parameterization. The hypothesis that will be tested is that the combination parameterization approach will achieve the best optimal solution and that the non-clustered will preform the worst. This will be verified with various number of robots printing a circular layer geometry.

In the case of two robots, the non-clustered method took the pre-determined maximum of 5000 generations and did not converge, the three parameterized methods took only a few hundred generations. The medial/radial combination and the radial strategies ended up at nearly identical optimal values, but

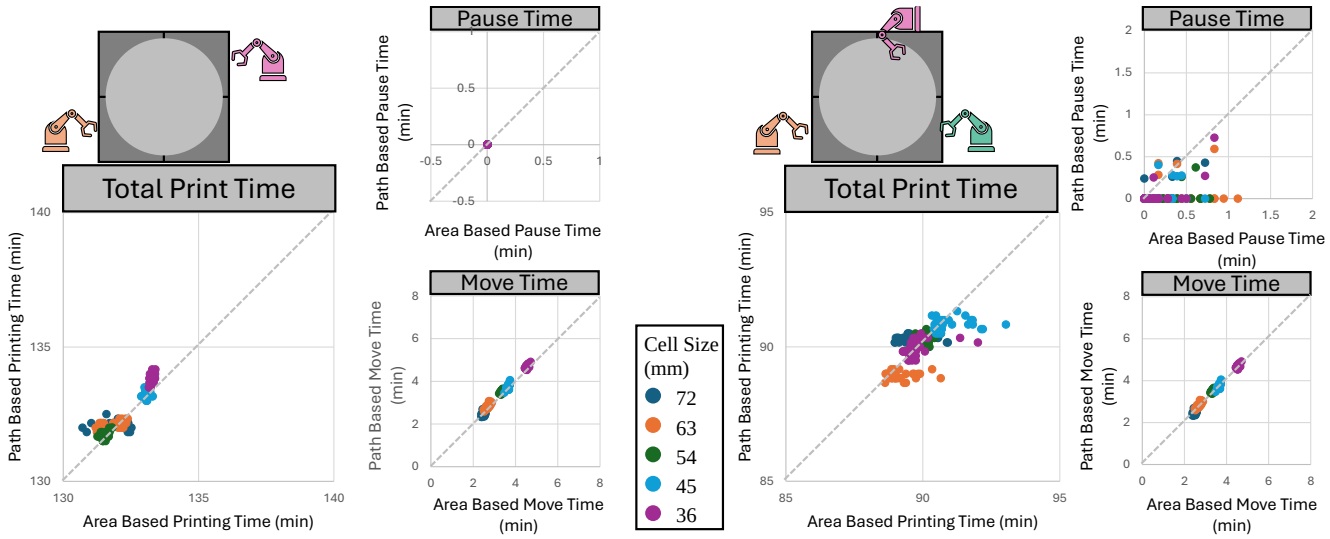
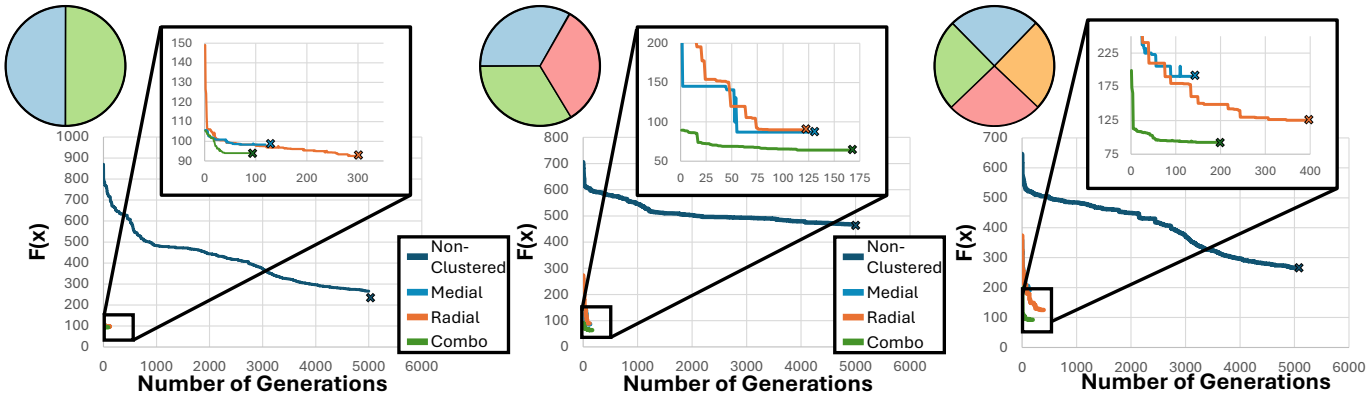


Figure 9: By computing starting the optimization process from 30 different starting populations we compare the time to print, move, and pause time using the area based (x-axis) and the path based (y-axis) with 2 and 3 robots which shows that they produce similar results. The dashed line on each graph is the ideal case if the two methods matched exactly.



a) 2 Robot Optimization b) 3 Robot Optimization c) 4 Robot Optimization

Figure 10: The convergence graph for different parameterization, radial only, medial only, and a combination of radial and medial all converge faster and to a better optimum than a non-clustered sequence.

the combination took 200 fewer generations (Fig. 10 a)). For three robots, the non-clustered does not converge (50 generations without change in optimism) at all, performing poorly. In this case, the combination reached a better optimum than the radial or medial parameterizations but took a few more generations (Fig. 10 b)). This trend continues for four robots, where the combination performs much better than the two parameterized models and the non-clustered takes the maximum of 5000 generations and did not converge as well as not reaching a value near the parameterized approaches (Fig. 10 c)).

From this testing, we can determine that the two parameterizations presented in this work perform much better than the non-clustered approach. Furthermore, combining these two parameterizations and allowing for the GA to change between them as needed generally results in better or equal results. While we only show with a single setup of geometry and a simple approach to robot positioning, future results reinforce this result.

5.4. Effect of Cell Resolution

The final numerical study conducted is to determine the effect of different cell sizes, which can vary depending on the tessellation of the printing part. Prior studies on C3DP show that changing the cell size from larger to small leads to an increase in printing time as the movement time of the robots also increases (Krishnamurthy et al., 2022; Ebert et al., 2025). However, these works have different collision avoidance protocols which our work does not have. The main question that we are exploring is whether smaller cells allow for better sequencing such that pausing occurs less often, despite a potential increase in move time. This test will be conducted on a large part of size 1.5m x 300mm with 5 robots in various orientations around the part (Fig. 11). The robot setup is based on the constraints of our physical robot setup. The cell size will be sampled near the value of the radius of the nozzle (40mm), which is the safe distance that the robots must maintain from each other. Although the parameterization test demonstrated that a combination of medial and radial is best, we will use the same four testing methods as before for completeness. We utilize several different orientations of robots to validate that the results are not due to a specific robot placement/positioning. To see further results, see the appendix (Supp. Fig. 1).

The first observation for the first orientation of robots is that, as the cell size decreases, the time to print increases for the non-clustered(Fig. 11 a). However, the medial, radial, and combination parameterization it generally decreases or stays the same as the cell size decreases. When the cell size decreases, the combination parameterization becomes the best by a wide margin, while non-clustered becomes the worst. After closer consideration this makes sense because with smaller cells there are more cells to sequence and non-clustered methods will have trouble converging while parameterized methods may be able to find sequences which do not have pause time.

The second orientation has a similar result. As the cell size decreases, the non-clustered becomes much worse in overall time to print, while the parameterized methods remain similar in overall print time. While the combination parameterization does not perform significantly better than the other methods, it is still the best in most cases.

Parameterization Method	Time to Print (min)
Slicer-based contour method	95.29
Cellularization without Optimization	91.66
Cellularization with Optimization	86.77

Table 2: The time taken for the different printing methods chosen in physical evaluation. The optimized cellular approach takes less time than the non-optimized cellular and the slicer-based contour method. Examples of the path are in Fig. 12.

Although this set of tests was done on a limited setup (e.g., geometry and number of robots), it demonstrates a result contrary to other cellularized work where smaller cells do not require a trade-off to higher print time. These results further demonstrate that overall parameterized clustering aids in finding a better optimal than a non-clustered methods.

6. Physical Validation

The goal of our physical testing is twofold. First, we aim to determine the impact of parameterization for printing. Secondly, we physically validate that the algorithm and sequencing strategy can be utilized sequentially, on separate layers, to slice and print a multi-layer object with multiple robots.

6.1. Parameterization

The first test consists of physically printing a banana shape with dimensions of approximately 850*mm* x 270*mm* using 3 robots. Three different methods for printing will be tested, namely:

1. Slicer-based: Using a standard slicer, contours are made from the outside of the part and working inside (Fig. 12 a).
2. Cellular Non-Optimized: The cellularization approach presented in this paper and all cells are parameterized using the medial approach, but the printing sequence is not optimized. The printing order is the identity where the first cluster is printed first, second cluster is printed second, etc. (Fig. 12 b).
3. Cellular Optimized: Similar to (b), however, in this case the sequence is determined by our GA optimizer, allowing some robots to use radial sequencing and others to use medial sequencing (Fig. 12 c).

These banana prints will be made using the robot setup described in prior work (Poudel et al., 2020a,b, 2022) and only a single layer will be printed (Fig. 14 a)).

The key result from the testing is that the circularized optimized result took the lowest overall time to print and was 8.5 minutes faster than the slicer-based method, and 4.89 minutes faster than the non-optimized approach. While this time saving may seem insignificant, as a percentage of the overall time to print, each robot has more than 43 minutes of extruding time

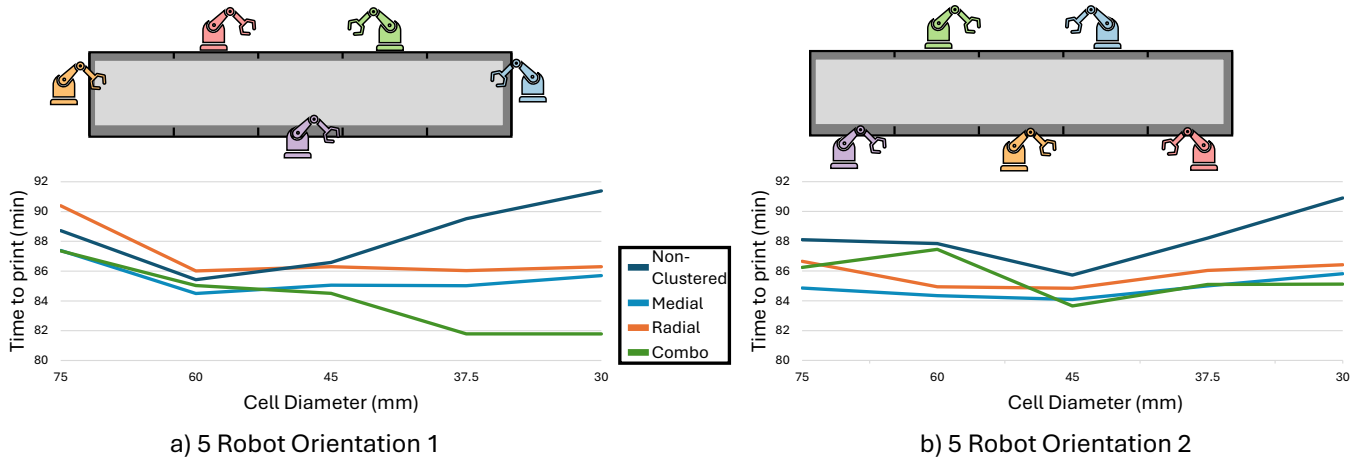


Figure 11: The time to print a large object (1500mm x 300mm) with 5 robots in two orientations is shown. As the cell size decreases, the time to print for the parameterized methods decreases or remains similar.

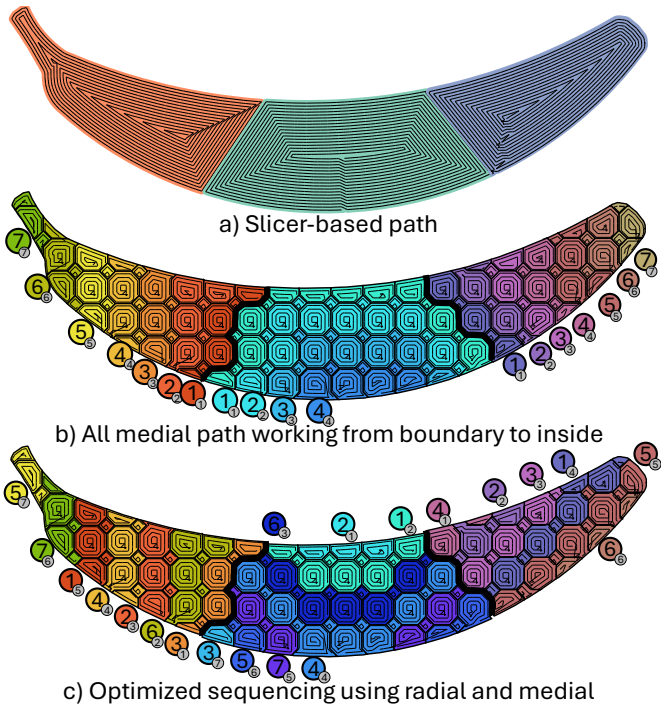


Figure 12: The printing path for the robots using different strategies: the slicer based approach a), the non-optimized identity cellular approach b), and the optimized cellular approach which contains medial and radial contours.

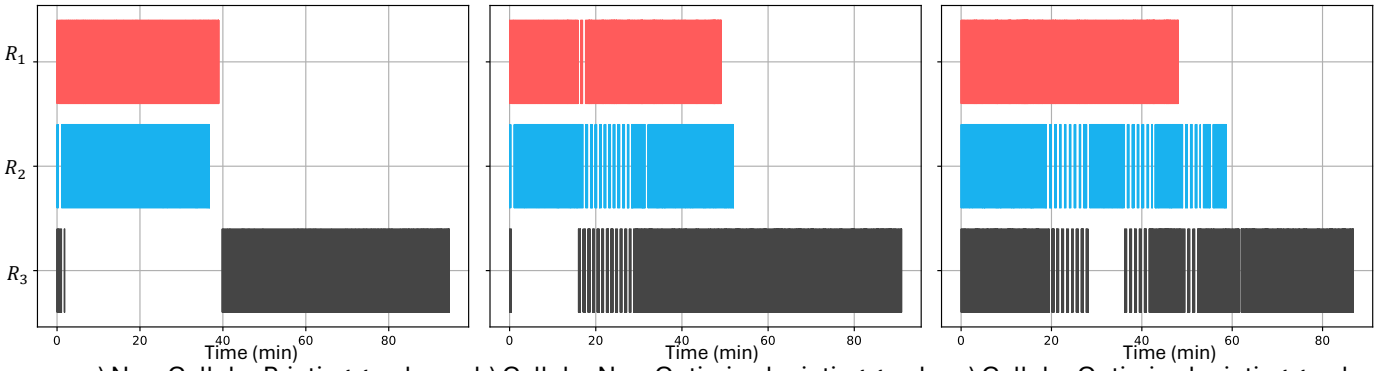
which cannot be reduced. Furthermore, if 8.5 minutes are saved every layer for 200 layers it would result in over 24 hours of reduced time to print.

The time each robot spends extruding can also be compared. In the case of the slicer-based the printing occurred almost sequentially where the robots printing in red and in blue printed their work in parallel and then the robot printing in black finished its work while no other robots were working (Fig. 13 a). This is much different from the other two cases where all three robots worked together for a significant portion of the print. In the non-optimized case, the robot printing in black has to wait until the red and blue robot are far enough away until it can start printing. In the case of the optimized all robots begin by printing together and then black robot pauses for a short time before resuming its work.

6.2. *Multi-Layer Printing*

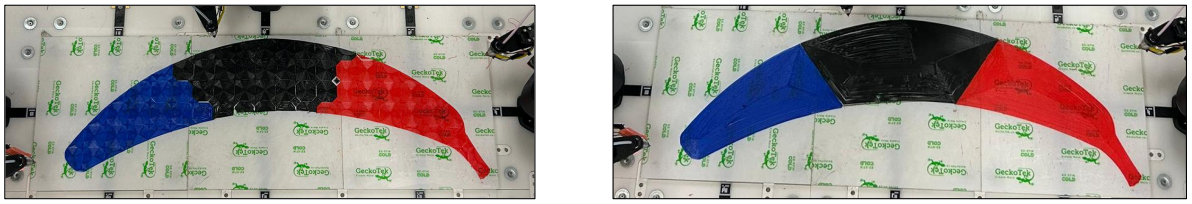
We further validate the algorithm by printing multi-layer objects. In order to complete the optimization on the objects, we first equally distribute the workload between robots by moving the part within the workspace, since workload changes between layers we sum the total work for each robot across all layers and move the part until work is equally distributed between robots. While this approach does not guarantee that all layers have equal distribution, it ensures equal work across all layers. This is also similar to the approach by NoodlePrint with their concurrence measure but without the penalty for interfacing cells (Ebert et al., 2025). The optimization can then be performed sequentially on every layer to determine the printing order for the clusters. Additionally, each robot may change between radial and medial, as shown in the chromosome representation.

A model of Einstein was chosen to print with approximate dimensions of 475mm x 290mm x 135mm. During printing, it was observed that the segmentation method using truncated octahedron the interfaces were always planar but changed direction every few layers (Fig. 14 b)).

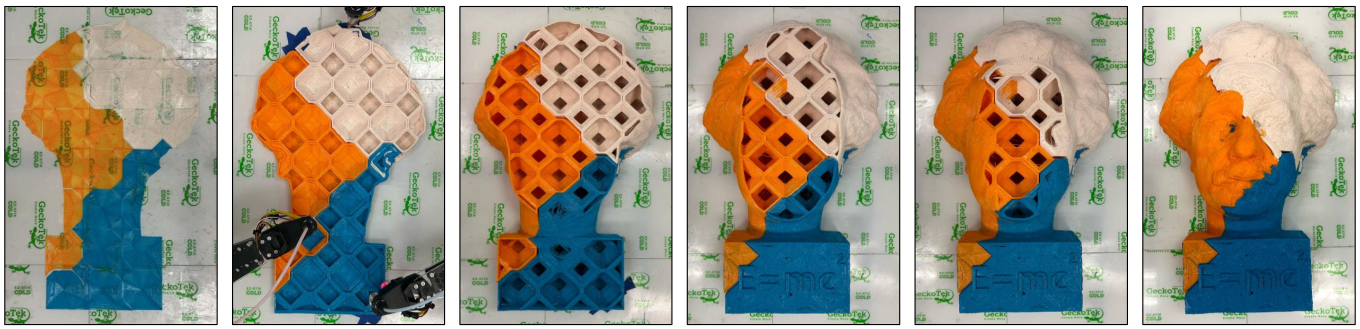


a) Non-Cellular Printing graph b) Cellular Non-Optimized printing graph c) Cellular Optimized printing graph

Figure 13: The times when extrusion occurred during the physical printing of the banana per robot (red, blue, black).



a) The physical prints of the banana cellular (left), and non-cellular (right)



b) Printing of an Einstein multi-layer object

Figure 14: The times when extrusion occurred during the physical printing of the banana per robot (red, blue, black).

7. Discussion

7.1. Effectiveness of Parameterization

One of the main results of this work is demonstrating that parameterization of the space allows for better sequencing of tasks to avoid collision and work in a time-optimal manner. This was first demonstrated with the numerical results, where the non-clustered performs poorly in overall time to print and struggles to converge on an optimal schedule (Section 5.3). The parameterized methods allow for much faster convergence. Additionally, using a combination of medial and radial parametrization allows for better performance than just using one method.

This was further demonstrated in the physical results, where the slicer-based method took the longest time to print, a naive non-optimized approach for printing using parameterization achieved a better time to print. The optimized parameterized approach took the least time to print and paused much less than the other methods.

7.2. Print Time Calculation

In order to reduce the computational effort to calculate total print time, the area-based evaluation method was introduced, which uses a conservative cell-to-cell distance calculation. The cell-to-cell distance can be precomputed and used throughout the optimization process, significantly reducing overall optimization time. It was also shown that computing the optimal using the path-based method was similar in overall time to print, idle time, and movement time compared to the path-based method. This result shows that future studies can reduce the computational effort when evaluating the fitness function by using the area-based method, which allows for more efficient optimization without compromising the accuracy.

7.3. Scalability and Practical Considerations

Other works in C3DP that use a safe-area collision avoidance algorithmic approach make all other robots pause while one robot is printing an unsafe region and have scalability difficulties since there is little time for all robots to work together (Krishnamurthy et al., 2022; Ebert et al., 2025). The Traveling Cellsman framework allows for all robots to start printing and working on their regions, while in some configurations printing cannot happen continuously, so the amount of pause time is minimized. This was demonstrated in the numerical tests, where it was shown that the cell size is relatively independent of the time to print since smaller cells have higher movement time but lower pause time due to better optimums. This result was unexpected since prior works have shown that smaller cells have higher print time due to higher movement time but no method to reduce the pause time (Krishnamurthy et al., 2022; Ebert et al., 2025). We also demonstrated with 3 robots that can print a large multi-layer object by conducting optimization on every layer in order to allow for all robots to collaborate during printing.

One of the observations from physical printing was that there were gaps between the pieces printed by different robots which can lead to weak interface (Ebert et al., 2025). These gaps could have been caused by initial calibration or compounding small errors in the robot motor positioning throughout the printing process. While using some methods of segmentation this can lead to disastrous results, using the truncated octahedron allowed for maintaining cohesion between the parts printed by different robots. This is due the angled interface between truncated octahedron constantly change direction (Fig. 14 b)). Additionally, the truncated octahedron creates cells on every layer which are either squares, or hexagons which can be much easier to print without harsh angles.

7.4. Other Multi-Robot Scenarios

While this work has focused on collaborative 3D printing, other multi-robot task scheduling applications could also be used with parameterization. As an example, colored MTSP exist in milling applications where multiple robots must work in shared spaces (Li et al., 2014a). In this way, the parameterization shown here can be used to sequence contours, where each contour is the bit size. Other potential applications can also include welding, WAAM 3D printing, painting/spraying, manipulation/assembly, or surface prep applications where multiple robots share a working space. In these cases, optimizing the scheduling between tasks is critical since inefficient movements can result in wasted time or inefficient use of resources.

8. Conclusion

This work has shown that using parameterized clustering can be useful in creating the schedule for printing cells in cooperative 3D printing tasks. In order to create clusters efficiently, we introduced radial and medial clustering, which are formed based on the partitioning of space and its subsequent cellularization. The numerical results showed that clustering allowed for faster convergence to an optimal solution as opposed to a non-clustered approach. Additionally, allowing for different clustering methods (radial or medial) per robot allowed for better solutions. Finally, the physical results validated that the optimization process reduces the overall time to print as compared to a non-optimized "moving-front" style algorithm as well a slicer-based algorithm, traditionally used in C3DP applications.

While we only showed the benefit of clustering for C3DP applications, this approach can be generally applicable to many different types of multi-robot applications where traversal between tasks must avoid collision while also minimizing total time moving between tasks. This is particularly relevant for multi-robot additive manufacturing applications, as our framework is agnostic to the process type (e.g., FDM or cold-spray).

Acknowledgment

This work was partially supported by the National Science Foundation (NSF) Awards #2048182 (Engineering Design and Systems Engineering Program) and #2112009 (Small Business Innovation Research Program). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

Alhijaily, A., Kilic, Z.M., Bartolo, A.N.P., 2023a. Teams of robots in additive manufacturing: a review. Virtual and Physical Prototyping 18, e2162929. URL: https://doi.org/10.1080/17452759.2022.2162929, doi:10.1080/17452759.2022.2162929.

Alhijaily, A., Kilic, Z.M., Bartolo, P., 2023b. Online cooperative printing by mobile robots. Virtual and Physical Prototyping 18, e2276257.

Alitappeh, R.J., Jeddisaravi, K., 2022. Multi-robot exploration in task allocation problem. Applied Intelligence 52, 2189–2211.

Arbogast, A., Nycz, A., Noakes, M.W., Wang, P., Masuo, C., Vaughan, J., Love, L., Lind, R., Carter, W., Meyer, L., et al., 2024. Strategies for a scalable multi-robot large scale wire arc additive manufacturing system. Additive Manufacturing Letters 8, 100183.

Bao, X., Wang, G., Xu, L., Wang, Z., 2023. Solving the min-max clustered traveling salesmen problem based on genetic algorithm. Biomimetics 8, 238.

Bektas, T., 2006. The multiple traveling salesman problem: an overview of formulations and solution procedures. omega 34, 209–219.

Bhatt, P.M., Kabir, A.M., Malhan, R.K., Shah, B., Shembekar, A.V., Yoon, Y.J., Gupta, S.K., 2019. A robotic cell for multi-resolution additive manufacturing, in: 2019 International conference on robotics and automation (ICRA), IEEE. pp. 2800–2807.

Bhatt, P.M., Nycz, A., Gupta, S.K., 2022. Optimizing multi-robot placements for wire arc additive manufacturing, in: 2022 International Conference on Robotics and Automation (ICRA), IEEE. pp. 7942–7948.

Carter, A.E., Ragsdale, C.T., 2006. A new approach to solving the multiple traveling salesperson problem using genetic algorithms. European journal of operational research 175, 246–257.

Chandran, N., Narendran, T., Ganesh, K., 2006. A clustering approach to solve the multiple travelling salesmen problem. International Journal of Industrial and Systems Engineering 1, 372–387.

Dörfler, K., Dielemans, G., Lachmayer, L., Recker, T., Raatz, A., Lowke, D., Gerke, M., 2022. Additive manufacturing using mobile robots: Opportunities and challenges for building construction. Cement and Concrete Research 158, 106772.

Ebert, M., Stone, R., Koithan, J., Zhou, W., Pharr, M., Estrin, Y., Akleman, E., Sha, Z., Krishnamurthy, V., 2025. Noodleprint: Cooperative multi-robot additive manufacturing with helically interlocked tiles. Journal of Manufacturing Science and Engineering , 1–21URL: https://doi.org/10.1115/1.4067617, doi:10.1115/1.4067617.

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

Fei, Y., Fuqiang, D., Xifang, Z., 2004. Collision-free motion planning of dual-arm reconfigurable robots. Robotics and Computer-Integrated Manufacturing 20, 351–357. URL: https://www.sciencedirect.com/science/article/pii/S0736584504000031, doi:https://doi.org/10.1016/j.rcim.2004.01.002.

Ganganath, N., Cheng, C.T., Fok, K.Y., Chi, K.T., 2016. Trajectory planning for 3d printing: A revisit to traveling salesman problem, in: 2016 2nd International Conference on Control, Automation and Robotics (ICCAR), IEEE. pp. 287–290.

Gombolay, M., Wilcox, R., Shah, J., 2013. Fast scheduling of multi-robot teams with temporospatial constraints. Robotics: Science and Systems IX .

Gul, F., Mir, A., Mir, I., Mir, S., Islaam, T.U., Abualigah, L., Forestiero, A., 2022. A centralized strategy for multi-agent exploration. IEEE Access 10, 126871–126884. doi:10.1109/ACCESS.2022.3218653.

Jiang, C., Wan, Z., Peng, Z., 2020. A new efficient hybrid algorithm for large scale multiple traveling salesman problems. Expert Systems with Applications 139, 112867.

Jin, Y., Pierson, H.A., Liao, H., 2019. Toolpath allocation and scheduling for concurrent fused filament fabrication with multiple extruders. IISE Transactions 51, 192–208.

Keshvarparast, A., Battini, D., Battaia, O., Pirayesh, A., 2024. Collaborative robots in manufacturing and assembly systems: literature review and future research agenda. Journal of Intelligent Manufacturing 35, 2065–2118.

Király, A., Abonyi, J., 2010. A novel approach to solve multiple traveling salesmen problem by genetic algorithm. Computational Intelligence in Engineering , 141–151.

Király, A., Abonyi, J., 2011. Optimization of multiple traveling salesmen problem by a novel representation based genetic algorithm, in: Intelligent Computational Optimization in Engineering: Techniques and Applications. Springer, pp. 241–269.

Kitjacharoenchai, P., Ventresca, M., Moshref-Javadi, M., Lee, S., Tanchoco, J.M., Brunese, P.A., 2019. Multiple traveling salesman problem with drones: Mathematical model and heuristic approach. Computers & Industrial Engineering 129, 14–30.

Kong, X., Gao, Y., Wang, T., Liu, J., Xu, W., 2019. Multi-robot task allocation strategy based on particle swarm optimization and greedy algorithm, in: 2019 IEEE 8th joint international information technology and artificial intelligence conference (ITAIC), IEEE. pp. 1643–1646.

Krishnamurthy, V., Poudel, L., Ebert, M., Weber, D.H., Wu, R., Zhou, W., Akleman, E., Sha, Z., 2022. Layerlock: Layer-wise collision-free multi-robot additive manufacturing using topologically interlocked space-filling shapes. Computer-Aided Design 152, 103392.

Kuipers, T., Su, R., Wu, J., Wang, C.C., 2022. Itil: Interlaced topologically interlocking lattice for continuous dual-material extrusion. Additive Manufacturing 50, 102495.

Larsen, L., Pham, V.L., Kim, J., Kupke, M., 2015. Collision-free path planning of industrial cooperating robots for aircraft fuselage production. 2015 IEEE International Conference on Robotics and Automation (ICRA) , 2042–2047doi:10.1109/ICRA.2015.7139466.

Lawler, E.L., 1985. The traveling salesman problem: a guided tour of combinatorial optimization. Wiley-Interscience Series in Discrete Mathematics .

Li, J., Sun, Q., Zhou, M., Yu, X., Dai, X., 2014a. Colored traveling salesman problem and solution. IFAC Proceedings Volumes 47, 9575–9580.

Li, J., Zhou, M., Sun, Q., Dai, X., Yu, X., 2014b. Colored traveling salesman problem. IEEE transactions on cybernetics 45, 2390–2401.

Liao, E., Liu, C., 2018. A hierarchical algorithm based on density peaks clustering and ant colony optimization for traveling salesman problem. Ieee Access 6, 38921–38933.

Lin, H.Y., Huang, Y.C., 2021. Collaborative complete coverage and path planning for multi-robot exploration. Sensors 21, 3709.

Liu, S., Gao, X., Chen, L., Zhou, S., Peng, Y., Yu, D.Z., Ma, X., Wang, Y., 2023a. Multi-traveler salesman problem for unmanned vehicles: Optimization through improved hopfield neural network. Sustainability 15, 15118.

Liu, S., Shen, J., Tian, W., Lin, J., Li, P., Li, B., 2023b. Balanced task allocation and collision-free scheduling of multi-robot systems in large spacecraft structure manufacturing. Robotics and Autonomous Systems 159, 104289.

Martin, J.G., Muros, F.J., Maestre, J.M., Camacho, E.F., 2023. Multi-robot task allocation clustering based on game theory. Robotics and Autonomous Systems 161, 104314.

Marvel, J.A., Bostelman, R., Falco, J., 2018. Multi-robot assembly strategies and metrics. ACM Computing Surveys (CSUR) 51, 1–32.

McPherson, J., Zhou, W., 2018. A chunk-based slicer for cooperative 3d printing. Rapid Prototyping Journal 24, 1436–1446.

Meng, X., Li, J., Dai, X., Dou, J., 2017. Variable neighborhood search for a colored traveling salesman problem. IEEE Transactions on Intelligent Transportation Systems 19, 1018–1026.

Nallusamy, R., Duraiswamy, K., Dhanalaksmi, R., Parthiban, P., 2010. Optimization of non-linear multiple traveling salesman problem using k-means clustering, shrink wrap algorithm and meta-heuristics. International Journal of Nonlinear Science 9, 171–177.

Pandiri, V., Singh, A., 2018. A swarm intelligence approach for the colored traveling salesman problem. Applied Intelligence 48, 4412–4428.

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

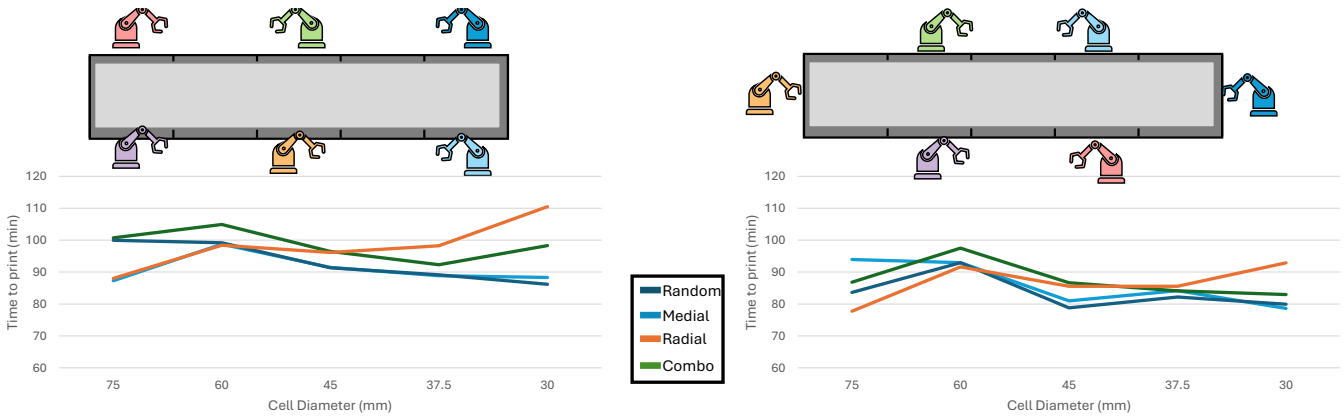
Poudel, L., Marques, L.G., Williams, R.A., Hyden, Z., Guerra, P., Fowler, O.L., Sha, Z., Zhou, W., 2022. Toward swarm manufacturing: Architecting a cooperative 3d printing system. Journal of Manufacturing Science and Engineering 144, 081004.

Poudel, L., Sha, Z., Zhou, W., 2018. Mechanical strength of chunk-based printed parts for cooperative 3d printing. Procedia Manufacturing 26, 962–972.

Poudel, L., Zhou, W., Sha, Z., 2020b. A generative approach for scheduling multi-robot cooperative three-dimensional printing. Journal of Computing

and Information Science in Engineering 20, 061011.
Poudel, L., Zhou, W., Sha, Z., 2021. Resource-Constrained Scheduling for Multi-Robot Cooperative Three-Dimensional Printing. Journal of Mechanical Design 143. URL: https://doi.org/10.1115/1.4050380, doi:10.1115/1.4050380. 072002.
Sahu, A., Panda, B., 2024. A tsp-based continuous path planning for additive manufacturing of concrete. Progress in Additive Manufacturing , 1–22.
Singh, S., Singh, A., Kapil, S., Das, M., 2022. Utilization of a tsp solver for generating non-retractable, direction favouring toolpath for additive manufacturing. Additive Manufacturing 59, 103126.
Stone, R., Ebert, M., Zhou, W., Akleman, E., Krishnamurthy, V., Sha, Z., 2025. Safezone*: A graph-based and time-optimal cooperative 3d printing framework. Journal of Computing and Information Science in Engineering , 1–14.
Stone, R.F., Ebert, M., Zhou, W., Akleman, E., Krishnamurthy, V., Sha, Z., 2024. Safezone: A topologically-aware voronoi-based framework for fast collision-free cooperative 3d printing, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, p. V02AT02A038.
Stone, R.F., Zhou, W., Akleman, E., Krishnamurthy, V.R., Sha, Z., 2023. Print as a dance duet: Communication strategies for collision-free arm-arm coordination in cooperative 3d printing, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, p. V002T02A081.
Sustarevas, J., Benjamin Tan, K.X., Gerber, D., Stuart-Smith, R., Pawar, V.M., 2019. Youwasps: Towards autonomous multi-robot mobile deposition for construction, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2320–2327. doi:10.1109/IROS40897.2019.8967766.
Tereshchuk, V., Stewart, J., Bykov, N., Pedigo, S., Devasia, S., Banerjee, A.G., 2019. An efficient scheduling algorithm for multi-robot task allocation in assembling aircraft structures. IEEE Robotics and Automation Letters 4, 3844–3851.
Weber, D.H., Zhou, W., Sha, Z., 2022. Z-chunking for cooperative 3d printing of large and tall objects, in: 2022 International Solid Freeform Fabrication Symposium.
Wei, C., Ji, Z., Cai, B., 2020. Particle swarm optimization for cooperative multi-robot task allocation: a multi-objective approach. IEEE Robotics and Automation Letters 5, 2530–2537.
Xidias, E.K., Zacharia, P.T., Aspragathos, N.A., 2010. Time-optimal task scheduling for articulated manipulators in environments cluttered with obstacles. Robotica 28, 427–440.
Zacharia, P.T., Aspragathos, N., 2005. Optimal robot task scheduling based on genetic algorithms. Robotics and Computer-Integrated Manufacturing 21, 67–79.
Zbiss, K., Kacem, A., Santillo, M., Mohammadi, A., 2022. Automatic collision-free trajectory generation for collaborative robotic car-painting. IEEE Access 10, 9950–9959.
Zbiss, K., Kacem, A., Santillo, M., Mohammadi, A., 2024. Automatic optimal robotic base placement for collaborative industrial robotic car painting. Applied Sciences 14, 8614.
Zhang, K., Chermprayong, P., Xiao, F., Tzoumanikas, D., Dams, B., Kay, S., Kocer, B.B., Burns, A., Orr, L., Alhinai, T., et al., 2022. Aerial additive manufacturing with multiple autonomous robots. Nature 609, 709–717.
Zhang, X., Li, M., Lim, J.H., Weng, Y., Tay, Y.W.D., Pham, H., Pham, Q.C., 2018. Large-scale 3d printing by a team of mobile robots. Automation in Construction 95, 98–106.
Zhang, Y., Parker, L.E., 2013. Multi-robot task scheduling, in: 2013 IEEE international conference on robotics and automation, IEEE. pp. 2992–2998.

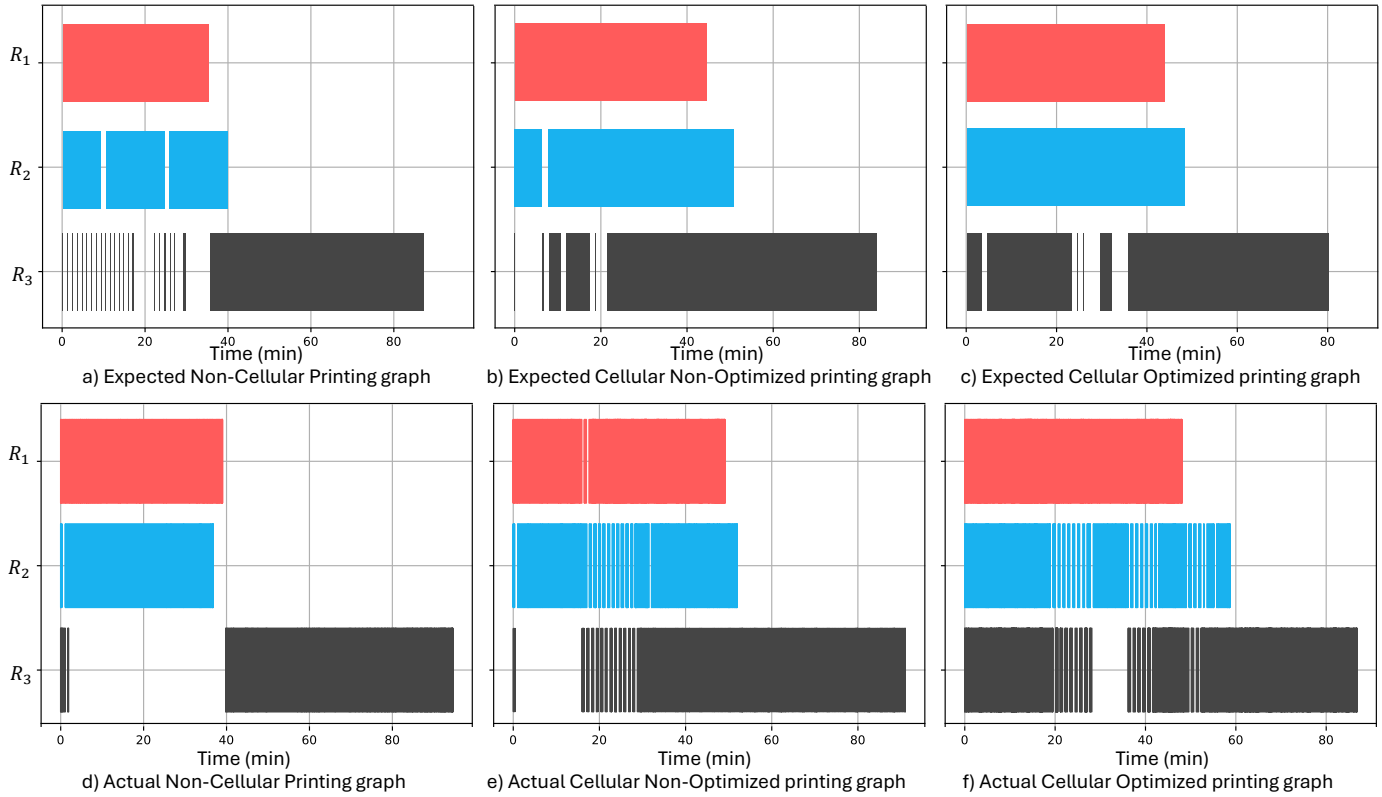
Appendix A. Supplemental Data



a) 6 Robot Orientation 1

b) 6 Robot Orientation 2

Supp. Fig. 1: Similar to Fig 11 the cell resultion was changed with a 6 robot setup and the effect on total printing time is shown.



a) Expected Non-Cellular Printing graph

b) Expected Cellular Non-Optimized printing graph

c) Expected Cellular Optimized printing graph

d) Actual Non-Cellular Printing graph

e) Actual Cellular Non-Optimized printing graph

f) Actual Cellular Optimized printing graph

Supp. Fig. 2: The expected times when printing should happen (top row) and the actual times printing occurred (bottom row). The bottom row is repeated from figure 13

Parameterization Method	Time to Print (min)	Avg time not extruding (min per robot)	Avg time extruding (min per robot)
Slicer-based contour method	95.29	51.96	43.20
Cellularization without Optimization	91.66	41.11	50.65
Cellularization with Optimization	86.77	34.60	50.62

Supp. Table 1: An extension of Tab. 2 which reports the total time taken by the robots pausing/moving/idling and extruding during physical evaluation. Examples of the path are in Fig. 12.

	2 Robots		3 Robots		4 Robots	
Parameterization Method	Optimal (min)	Num. Generations	Optimal (min)	Num. Generations	Optimal (min)	Num. Generations
Random	266.06	5000	467.088	5000	266.06	5000
Medial	98.332	119	86.92	165	190.48	139
Radial	92.548	331	89.922	136	125.35	397
Combo	93.99	89	64.125	166	92.556	196

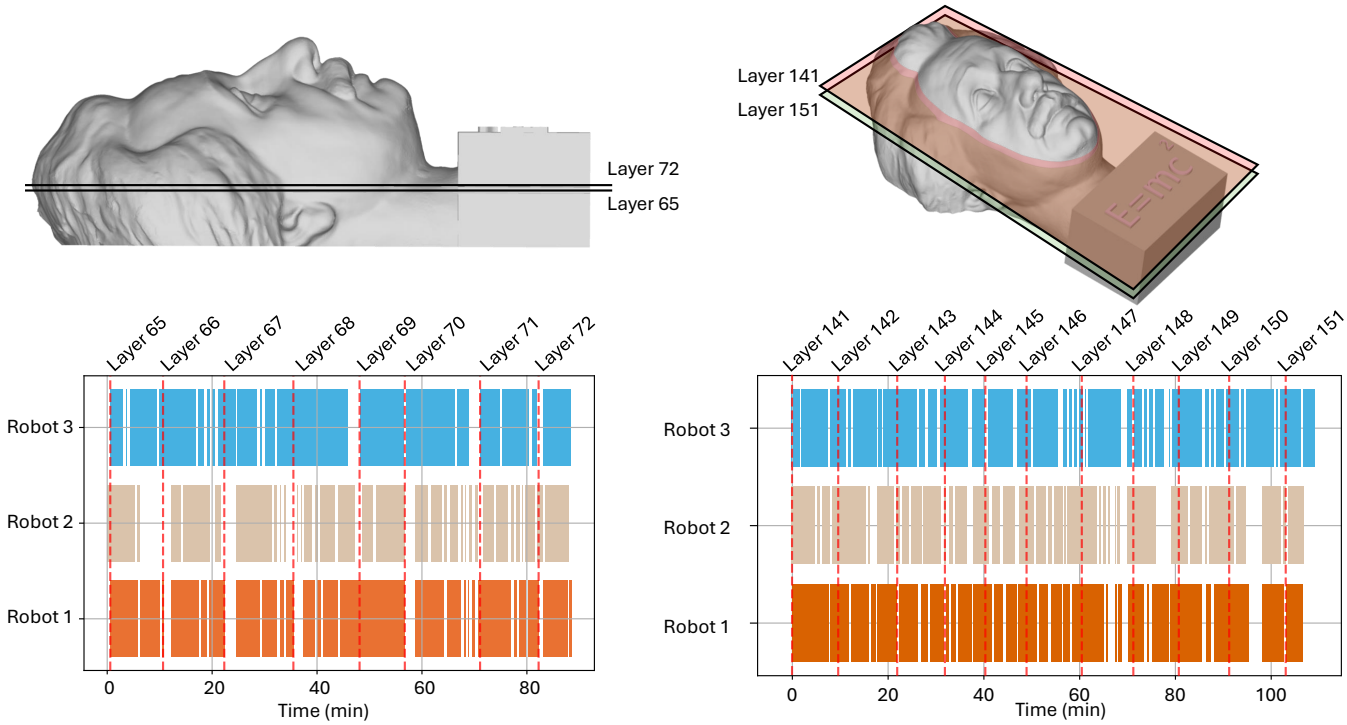
Supp. Table 2: The data from Fig. 10 which reports the optimal values and the number of generations to reach the optimal for the different parameterization methods, the maximum number of generations was set at 5,000.

	Cell Size (mm)	75	60	45	37.5	30
Random	Time (min)	88.1	87.8	85.7	88.2	90.9
Random	Num. Gen.	1237	3141	4809	5001	1000
Medial	Time (min)	84.8	84.3	84.1	85.0	85.8
Medial	Num. Gen.	133	134	172	203	343
Radial	Time (min)	86.6	84.9	84.8	86.0	86.4
Radial	Num. Gen.	214	400	827	594	700
Combo	Time (min)	86.2	87.4	83.6	85.1	85.1
Combo	Num. Gen.	157	210	238	230	205

Supp. Table 3: The time to print and the number of generations for Fig. 11 a

	Cell Size (mm)	75	60	45	37.5	30
Random	Time (min)	88.714	85.425	86.58	89.517	91.387
Random	Num. Gen.	1008	2122	5224	4465	6731
Medial	Time (min)	87.388	84.499	85.062	85.022	85.697
Medial	Num. Gen.	152	107	127	485	256
Radial	Time (min)	90.391	86.013	86.294	86.034	86.293
Radial	Num. Gen.	109	263	397	361	974
Combo	Time (min)	87.359	85.032	84.504	81.793	81.785
Combo	Num. Gen.	120	205	276	365	371

Supp. Table 4: The time to print and the number of generations for Fig. 11 b



a) Cooperation between layer 65 and 72

b) Cooperation between layer 141 and 151

Supp. Fig. 3: During the printing of the Einstein model, the time robots spent extruding was tracked, showing how well robots collaborate.