Ronnie F. P. Stone
Student Member ASME
Walker Department of Mechanical Engineering,
University of Texas at Austin,
Austin, TX 78712, USA
email: ronnie.stone@utexas.edu

Matthew Ebert
J. Mike Walker '66 Department of Mechanical Engineering,
Texas A&M University,
College Station, TX 77840, USA

Wenchao Zhou
Department of Mechanical Engineering,
University of Arkansas,
Fayetteville, AR 72701, USA

Ergun Akleman
Visual Computing & Computational Media,
Department of Computer Science and Engineering,
Texas A&M University,
College Station, TX 77840, USA

Vinayak Krishnamurthy
J. Mike Walker '66 Department of Mechanical Engineering,
Department of Computer Science and Engineering (Affiliated),
Texas A&M University,
College Station, TX 77840, USA

Zhenghui Sha[1]
Member ASME
Walker Department of Mechanical Engineering,
University of Texas at Austin,
Austin, TX 78712, USA
email: zsha@austin.utexas.edu

SafeZone*: A Graph-Based and Time-Optimal Cooperative 3D Printing Framework

Swarm manufacturing (SM) is an emerging manufacturing paradigm that employs a heterogeneous swarm of robots to accomplish complex hybrid manufacturing tasks. Cooperative 3D Printing (C3DP), a specialized form of swarm manufacturing, enables multiple printers to collaboratively produce large-scale parts, addressing key trade-offs in additive manufacturing, such as size, speed, quality, and cost. A fundamental challenge in C3DP is ensuring collision-free, time-optimal printing in a shared workspace. This is a complex problem that can be influenced by factors such as the number of printers, part geometry, printer positioning, mobility, and kinematics. In this paper, we present SafeZone, a collision-free and scalable C3DP framework that optimizes printing time by co-considering the geometry (area and shape) and topology (space-connectivity) of a shared workspace during layer partitioning. We first establish a conceptual framework to mathematically represent the topology of a layer through partition graphs. Then, we use a Voronoi tessellation within a constrained optimization framework to control the partition graph and minimize makespan. The Voronoi sites are associated with printer locations, allowing the framework to integrate physical constraints and facilitating solutions for systems with robotic manipulators. Physical testing in a four-printer scenario with robotic arms confirms that SafeZone* enables collision-free printing, resulting in a printing time reduction of 44.63% when compared to the single-printer scenario. Finally, numerical studies reveal trends in the optimal solutions concerning the chromatic number of their resulting partition graphs and the distribution of the printing areas among printers.*

Keywords: Cooperative 3D printing, Additive Manufacturing, Robotic Manipulators, Swarm Manufacturing.

1 Introduction

Swarm manufacturing (SM) has garnered significant attention in recent years for its potential to transform traditional manufacturing processes by employing a swarm of robots to cooperate on manufacturing tasks, particularly in the field of additive manufacturing (AM) [1,2]. Cooperative 3D Printing (C3DP), a specialized form of swarm manufacturing tailored to AM, uses multiple 3D printers (e.g., fused deposition modeling (FDM) printers) to work on a part simultaneously [3]. The printers can be either stationary or mobile, depending on the system configuration and desired application, but they typically operate in a shared workspace. Although working in a shared workspace is necessary to ensure cooperation and leverage the benefits of SM, it introduces two new challenges, namely division of labor and collision avoidance. In the context of C3DP, division of labor is concerned with how to divide a part into subvolumes and then assign them to each printer. Collision-avoidance refers to the ways (e.g., through control, scheduling, communication, detection) that the printers cooperate without colliding either with each other or with the part being printed, which could occur due to a printer's nozzle geometry or kinematic chain, as in the case of C3DP with robotic manipulators. The approach to solving these problems also heavily depends on the dynamics of the printing robots, since C3DP has been achieved with drones [4], mobile robots [5], and robotic manipulators [6].

Minimizing the overall printing time (i.e., makespan) is a standard objective in C3DP [7]. Due to the inherent interdependencies between the different stages of C3DP, an optimal framework must co-consider division of labor and collision avoidance, ensuring that each informs the other. For instance, if work is evenly distributed among all robots, but the collision-avoidance methodology is simplistic, the process may incur unavoidable collisions. Conversely, if the collision-avoidance strategy is robust and accommodates any division of labor, it may lead to suboptimal printing times. Further complexity is added to the problem when physical constraints are enforced. Such constraints are particularly relevant in systems with

[1]Corresponding Author.
Version 1.18, June 29, 2025

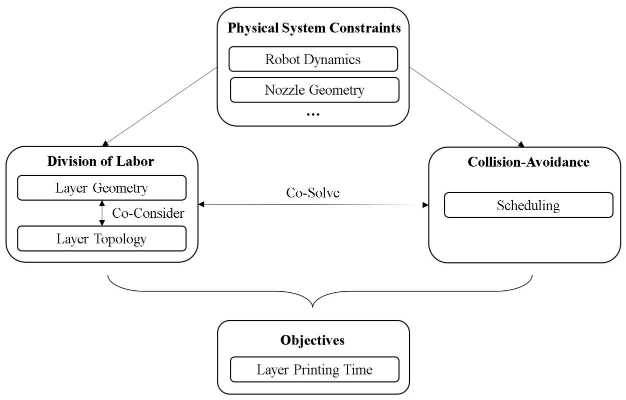

Fig. 1 Overall description of C3DP systems considering the key challenges of division of labor and collision-avoidance. SafeZone* co-considers layer geometry and topology and co-solves division of labor and collision-avoidance.

robotic manipulators, which may have issues with reachability and positioning of the printers in the workspace.

Existing C3DP frameworks primarily focus on the geometric aspects of the division of labor (i.e., size of sub-volumes) when minimizing printing time, often by attempting to evenly distribute labor among robots. However, this approach overlooks the topological aspects of the division of labor (i.e., connectivity of sub-volumes), which can significantly impact scheduling and collision avoidance. In our previous work, we demonstrated the importance of integrating the topology of the division of labor into the design space when optimizing C3DP frameworks [8]. Building on this foundation, we introduce SafeZone* (SZ*), a Voronoi- and graph-based C3DP framework for collision-free, minimum-makespan printing. The key idea of SZ* is to abstract the complexity of all C3DP stages by formulating an optimization framework that depends solely on printer positioning. By establishing a structured mathematical relationship between printer placement, layer partitioning (geometry and topology), and collision-free scheduling, SZ* efficiently searches the design space to identify C3DP configurations that minimize makespan.

SZ* is novel in that it minimizes makespan by co-considering the geometric and topological implications of the division of labor (Fig. 1). Particularly, SZ* demonstrates how an integrated topological-geometric analysis, which relates Voronoi tessellations to partition graphs, enables not only the efficient generation of optimal printing schedules, but also fundamental insights into beneficial topologies for C3DP, based on graph theory. Although SZ* is applicable to generic C3DP systems, it is tailored to robotic manipulator applications, where reachability and printer positioning are the primary concerns. Finally, SZ* is a hybrid approach: while the printing path and schedule are precomputed, execution is coordinated in real-time by a centralized control hub that sends commands to the printers. As such, the main contributions of this work are as follows:

(1) We establish a mathematical framework for representing and evaluating C3DP layer topology based on graph theory, with a focus on graph coloring.
(2) We introduce SZ*, a novel C3DP framework for makespan minimization that co-considers geometry and topology of a layer by relating Voronoi sites to partition graphs.
(3) We numerically validate SZ* and establish the chromatic number as a key metric for the topology of a minimum-makespan C3DP partition.
(4) We physically validate SZ* and demonstrate its robustness in constrained manufacturing environments by leveraging virtual robot positions.

The remainder of the paper is structured as follows. Section 2 reviews related research on C3DP and multi-robot systems that face similar problems. In Section 3, we establish the mathematical framework defining layer geometry, layer topology, and the key concept of buffer regions. The algorithmic framework of SZ* for achieving collision-free and minimum-makespan printing is presented in Section 4. In Section 5 we show the numerical results using the framework in a variety of scenarios, including different numbers of printers and layer geometries. Additionally, we present a case study with 4 SCARA printers as physical validation. Finally, we discuss limitations, future research directions, and topics of interest in Section 6.

2 Related Work

2.1 Collaborative Robots. While there has been extensive research on additive manufacturing [9], the concept of C3DP is a recent development [10,11]. There are several key challenges within C3DP that are not considered in additive manufacturing or robotics in general. With traditional single-robot or gantry-based 3D printing, collision avoidance between robots is not a concern. In many robotic tasks where multiple robots work together to complete a task, it is often preferable for robots to work as independently as possible, with little to no overlap in workspaces [12–14]. This can be seen in many coverage tasks where robots are required to cover the entire space, commonly through discovery and exploration [14,15]. In many cases, it is preferable that the robots interact as little as possible. In a work by Lin et al., two robots explore an unknown space, and in the ideal scenario, the paths of the robots never cross, and the robots, in general, have very little interaction with one another. A similar study demonstrates that efficient exploration of a known space can be achieved through the cellularization of the space, where each robot explores its own assigned region [16]. This is similar to what is done in C3DP, where robots are assigned known regions, and the method to segment the part into regions involves equal area distribution and, thus, nearly equal printing time. This approach is also seen in similar coverage works, such multi-robot car painting. In a work by Zbiss et al., several multi-axis robots work together to paint a car simultaneously. In this case, robots have shared workspaces, and collision avoidance is managed through careful ordering of the painting paths [17].

2.2 Collision Avoidance. When multiple robots operate in a shared environment, preventing collisions is essential. While various methods exist, we explore a few relevant to our work including, scheduling, zone-based approaches, and real-time collision detection.

By carefully scheduling printing tasks, collision can be avoided before the printing process begins. This approach has been used in C3DP by scheduling printing paths, [18,19] or sub-volumes [20–22]. In these cases, the relationship between part geometry and printing order must be carefully considered, as an overhanging region cannot before the region beneath it. One previously developed method uses dependency trees to determine printing order by leveraging the interfacial boundary of segments to construct the dependency tree [23]. The use of mobile robots and even aerial robots introduces additional dependencies, requiring careful coordination of printing sequences.

Another method for avoiding collisions is to classify regions as safe or unsafe, meaning that if printing occurs in one area, then all other robots either cannot print at all or are restricted to printing in specific regions [24–27]. While different papers use slightly varying terminology, the core idea remains the same: define regions where collisions are possible, typically along boundaries shared by multiple robots. These unsafe regions can then be managed by either printing them sequentially or restricting printing to a single robot at a time. In the case of Shen et al. [27], the division of labor was determined by optimizing the placement of cuts in an asymmetric part to evenly distribute work. Using this segmentation, an offset of the interfacial boundary was created to separate safe and unsafe regions [26,27]. Other approaches rely on predefined cellularization to define safe regions, where certain cells are labeled as safe or unsafe. In these methods, printing occurs sequentially in unsafe regions before transitioning to collaborative printing in safe regions (referred to as interfacing or non-interfacing in their terminology) [24,25].

Real-time collision avoidance can also be employed in C3DP, where robots continuously relay their current positions and/or trajectories to other robots. If a collision is imminent, one or more robots halt their work and resume once it is safe to proceed [26]. In such cases, optimizing the printing process becomes challenging since the path may not be predetermined, particularly when using a standard slicer for path generation. These methods can be beneficial in scenarios where the printing time or task completion time is unknown or variable, as other robots can continue working even if one is temporarily halted. However, scaling these methods to an arbitrary number of robots is difficult, as communication overhead and real-time distance calculations become increasingly costly with a larger number of robots.

While this is a brief overview of existing division of labor and collision-free scheduling methods, SZ* takes a fundamentally different approach by actively driving the partitioning process to ensure an optimal schedule. Unlike prior works that derive printing schedules from predefined partitions, SZ* simultaneously optimizes partitioning and scheduling through an integrated geometric-topological analysis of the C3DP layer. By leveraging Voronoi tessellations and partition graphs, it intelligently defines safe and unsafe regions while structuring them to enable collaborative, rather than strictly sequential, printing. This key distinction allows SZ* to minimize makespan while ensuring collision-free multi-robot coordination, setting it apart from existing methods.

3 Conceptual Framework

In this section, we formulate the problem of interest. Given a layer geometry, in the form of boundary curves, and a set of printers, the question is: how to partition the layer geometry into sub-regions and assign them to each printer? Ideally, the choice of partition should facilitate collision-free movement of the printers and, consequently, minimize printing time. While the *geometry* of each sub-region (i.e., size and shape) is clearly an important aspect to be determined, SZ* also considers the *topology* of the partitioning (i.e., how the sub-regions are connected to each other). In fact, current literature shows that the partitioning topology of robots working in a shared workspace is understudied.

To better understand the implication of the topology of the layer partitioning, particularly in the context of C3DP, let us consider a layer with a circular boundary that is set to be printed by four printers. A symmetric partition would be a simple, yet naive way to create the sub-regions (Fig. 2(a)). In this case, the likelihood of collision is high as we approach the center of the circle since that neighborhood is shared among all four printers. However, we observe that there may be other partitions of the same circular layer such that only three robots share any existing boundary (Fig. 2(b)). In fact, it is even possible to create sub-regions, such that no more than two robots share a boundary (Fig. 2(c)). Knowing that we are interested in achieving time-optimal and collision-free cooperative printing, these fundamental differences between each of the aforementioned partitions reveal the criticality of assessing their topology and geometry in conjunction. Hence, the goal of SZ* is to tackle these two aspects simultaneously.

3.1 The Topology of Partitioning. The partition topology of any layer geometry can be represented as a graph G_P, denoted as a *partition graph*, which is algebraically represented using an adjacency matrix M_P. The nodes of G_P represent the sub-regions, and the edges between the nodes represent the boundaries between sub-regions. Two sub-regions are said to be connected if they share a boundary, which can be a curve or a single point (Fig. 3(a)-(c)).

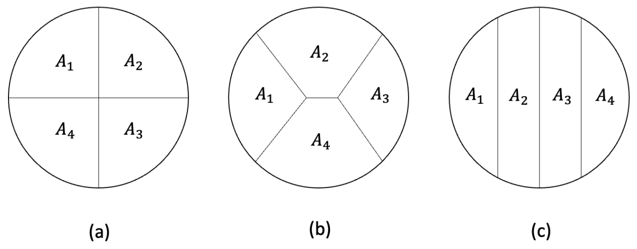

Fig. 2 4-Partitions of a circular layer geometry with different topologies. Each partition has progressively fewer shared boundaries. A boundary may consist of a single point.

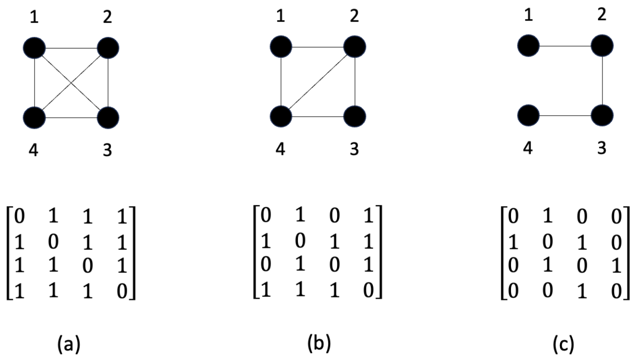
1
2
3
4
$\begin{bmatrix}0&1&1&1\\1&0&1&1\\1&1&0&1\\1&1&1&0\end{bmatrix}$
$\begin{bmatrix}0&1&0&1\\1&0&1&1\\0&1&0&1\\1&1&1&0\end{bmatrix}$
$\begin{bmatrix}0&1&0&0\\1&0&1&0\\0&1&0&1\\0&0&1&0\end{bmatrix}$
(a)
(b)
(c)

Fig. 3 Partition graphs and corresponding adjacency matrices of circular layer topologies. Nodes represent sub-regions and edges represent boundaries between sub-regions.

Generally, we can see that graphs that have nodes with lower degrees are better, in the sense that robots share fewer boundaries. However, we are interested in a more quantitative metric for judging whether a partition, topologically, is better than another. We approach this problem by relating the partition graph to the *printing schedule*, which, in turn, defines the printing time.

3.1.1 Goodness of Partition. Once a partition is chosen, the next logical question is: how do we order the printing of the sub-regions? If we assume that the printers can work uninterruptedly on their respective sub-regions, then the optimal partition would be any that divides the layer geometry into n equal-area sub-regions, where n is the total number of printers. Consequently, the optimal printing schedule is to print all regions in parallel. If we let T_l be the layer printing time, $T(X)$ the time taken to print sub-region X, and T_s the layer printing time when using a single printer, then:

$$T_l = \max(T(A_1),\ T(A_2),\ T(A_3),\ T(A_4)) = \frac{1}{4}T_s \tag{1}$$

In this idealized scenario, topology becomes irrelevant, and geometry dominates the choice of partition. To consider a more realistic scenario, let us assume that, if two sub-regions share a boundary, their respective printers cannot work on them simultaneously. As shown in our literature review, this is a common approach for achieving collision-free printing [26,27]. Under this assumption, we notice that certain layer topologies can offer parallel printing opportunities. For example, if two sub-regions do not share a boundary, then they can be printed completely in parallel without concern of collision. This is where we can leverage the concept of graph coloring [28] to guide us in drawing fundamental conclusions about the goodness of the topology of a partition.

3.1.2 Partition Graph Coloring. Let printer P_i be responsible for area A_i. Let S be the printing schedule, which we define as

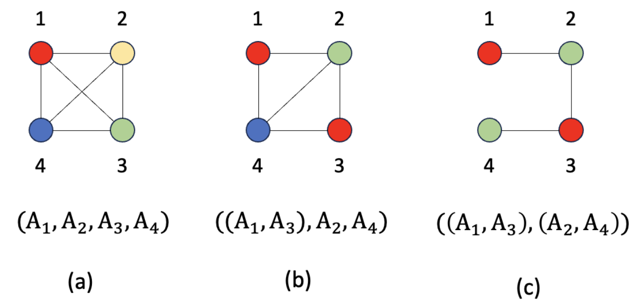

Fig. 4 Optimal printing schedule for different partitioning topologies using graph coloring on their respective partition graphs.

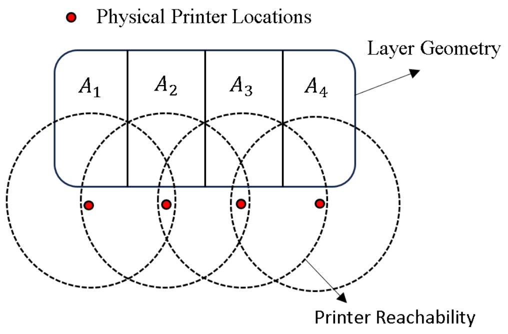

Fig. 5 General reachability issue in C3DP applications with robotic arms for degree-2 topologies. In this case, the partition graph is a path graph P_4.

a sequence of tuples τ_j. Each τ_j may consist of one or more A_i which can be printed in parallel and do not share edges with one another. If a τ_j has multiple A_i, it simply means that those regions are printed in parallel during that step of the schedule. For instance, in the idealized scenario we just discussed (Eq. 1), S = (τ_1), where $\tau_1 = (A_1, A_2, A_3, A_4)$, meaning that everything is parallelized.

Assuming that we are given a generic partition, we can represent it as a partition graph G_P and use any graph coloring technique to assign a color to each node such that no adjacent nodes have the same color. The coloring process is the same as determining a collision-free printing schedule. Thus, in our context, each color represents a tuple τ_j in the schedule S. The number of nodes with a given color represents the size of τ_j, which is the number of sub-regions printed in parallel at the j_{th} step of S.

When the number of nodes of the partition graph G_P is equal to the number of printers, meaning that each printer is responsible for printing only one sub-region, then it is straightforward to see that the optimal printing schedule will always use the variation of G_P with the smallest number of colors (Fig. 4(a)-(c)). To see why this is the case, let us first consider the worst case scenario, that is, the partition graph G_P has order N, and N is the minimum number of steps (colors) in its schedule. In this case, the layer printing can be written as:

$$T_l = \sum_{i=1}^{N} T(A_i) = T_s \tag{2}$$

meaning that the printing of each sub-region must be done sequentially to avoid collisions. This is clearly no better than using a single printer, with associated printing time of T_s, due to the lack of parallelization potential. However, assume now that the graph G_P has order N, but can be colored using $N - k$ colors, $k \in \mathbb{Z}^+$, $k < N$. In this new scenario, the layer printing time formula can be modified to:

$$T_l = \sum_{j=1}^{N-k} \max(T(\tau_j)) \tag{3}$$

where $T(\tau_j)$ is the time taken to print all the sub-regions contained in τ_j. As k approaches $N - 1$, which is related to using fewer colors, and we keep the order of the graph constant, the number of terms in the summation (Eq. 3) decreases. Consequently, the size of the tuples τ_j increases as more options for parallel printing become available. Thus, given that the area of the union of the sub-regions is constant since the layer geometry itself does not change, a lower number of summations term (i.e., more parallelization) will necessarily produce a better layer printing time, provided that the areas of the sub-regions A_i are chosen optimally. Otherwise, it is of course possible to construct scenarios in which a partition graph with a lower number of colors has a higher printing time by selectively picking poor choices of A_i. For example, this can be done by purposefully grouping larger A_i with smaller A_i in the tuples τ_j, increasing the makespan regardless of the number of τ_j.

In graph theory, the smallest number of colors needed to color a graph G such that no adjacent nodes share the same color is called the *chromatic number* of G, denoted $\chi(G)$. Therefore, according to our analysis of Eq. 3, the partition graph G_P with the smallest $\chi(G_P)$ is preferred. This is because the chromatic number determines the minimum number of steps in the printing schedule. If we consider the symmetric partition schedule (Fig. 4(a)), we see it has $\chi(G_P) = 4$, and so its layer printing time is the same as given by Eq. 2, which is no better than using a single printer. The last partition schedule (Fig. 4(c)), on the other hand, has $\chi(G_P) = 2$, resulting in a layer printing time of:

$$T_l = \max(T(A_1), T(A_3)) + \max(T(A_2), T(A_4)) = \frac{1}{2}T_s \tag{4}$$

assuming that the area of all sub-regions are equal, yielding half the printing time of using a single printer, which is a clear improvement over the case with $\chi(G_P) = 4$.

3.1.3 Degree-3 Partitions. We established, based on the printing schedule, and assuming that the order of the graph is equal to the number of printers, that the optimal topology generates a partition graph G_P with the lowest possible chromatic number $\chi(G_P)$. From graph theory, we know that, when considering the family of all possible n-partitions for connected geometries, $\chi(G_P)$ is bounded between 2 and n. If $\chi(G_P)$ is equal to n, it means that the resulting partition graph is complete (K_n). This is equivalent to the symmetric case (Fig. 4(a)), which produces the worst possible schedule for n printers (Eq. 2). If $\chi(G_P)$ is equal to 2, G_P becomes a path graph (P_n), which is equivalent to the third example provided (Fig. 4(c)).

Although topologies with P_n partition graphs have the smallest chromatic number, which would make them theoretically optimal from a printing scheduling perspective, they are generally not viable in C3DP systems with robotic manipulators due to scalability and reachability issues. These are two key constraints that SZ* considers in the formulation of its partitioning framework (Section 4.1). For instance, such a topology could produce sub-regions that stretch very far from the original positions of the printers. Additionally, as more printers are added, and the part of interest is not slender, the workspace may become too cluttered as the distances between the robot positions decreases (Fig. 5). However, there are special cases in which we can leverage other families of partition graphs that have $\chi(G_P) = 2$, in feasible ways (Section 6.1) but they are difficult to generalize to any layer geometry.

Therefore, in general, we are interested in finding the next best partitions which have $\chi(G_P) = 3$ (Fig 4(b)). This is only possible if we have, what we define to be a *degree-3 partition*. A degree-3

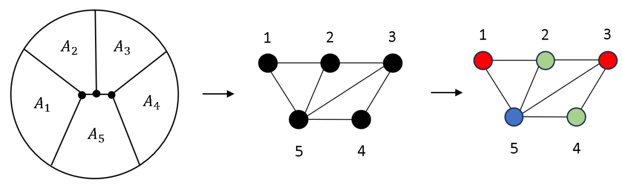

Fig. 6 Degree-3 partitions produce triangular grid partition graphs, which are guaranteed to be 3-colorable.

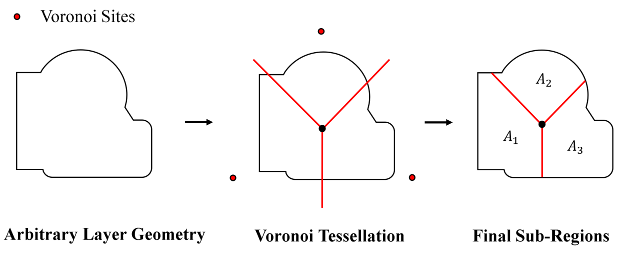

Fig. 7 Voronoi tessellation to partition the layer geometry into sub-regions.

partition is one in which the boundary curves connecting the sub-regions must share a degree-3 vertex. This condition ensures that the topology will always generate a partition graph G_P that is a triangular grid graph, which by definition is 3-colorable (Fig. 6). Note that when we say degree-3 partitions we are referring to the vertices of the actual physical boundaries of the tessellation and not the nodes of the partition graph G_P. In such a case, the layer printing time becomes:

$$T_l = \max(T(\tau_1)) + \max(T(\tau_2)) + T\max(T(\tau_3)) \quad (5)$$

where each τ_j represents one of the three steps (colors). Therefore, we can conclude that for the majority of C3DP applications with robotic manipulators, degree-3 partitions are preferable.

3.2 The Geometry of Partitioning. As such, there are several ways in which a layer geometry can be partitioned to allow for a desired topology. However, there are three main concerns to be addressed specifically for C3DP with robotic manipulators. First, we need to minimize the likelihood of collision between robots that share a common boundary. Second, the method should provide a means to avoid reachability issues. Finally, the method to generate the partitions should offer a natural way to control the partition topology, especially the degree-3 condition. Therefore, a systematic way of determining and controlling the geometry of the sub-regions A_i is necessary.

We note that constructing a Voronoi tessellation of the planar domain using the robot locations as the Voronoi sites naturally alleviates all of these concerns (Fig. 7). Given a set of points $\{p_i\} \in \mathbb{R}^n$, a Voronoi tessellation is a mutually exclusive and exhaustive set of polygonal regions $\{A_i\}$ such that any point $q \in A_i$ is closest to p_i. If $\{p_i\}$ are robot positions, then using such a partition to segment the layer geometry will automatically result in printable sub-regions that are also closest to the nearest robot. This addresses the first two concerns, namely minimizing the likelihood of collision between robots that share a common boundary in the partition and naturally avoiding reachability concerns.

Besides the closest-site property, Voronoi tessellations offer a natural way to control the topology of the partitioning at the vertices. To understand this, we consider the Delaunay diagram, which is the dual of the Voronoi tessellation (i.e. the vertices of Delaunay diagram are the sites of the Voronoi tessellation). The Delaunay diagram can consist of convex polygons that can all be triangulated to obtain the well-known Delaunay triangulation. Also note

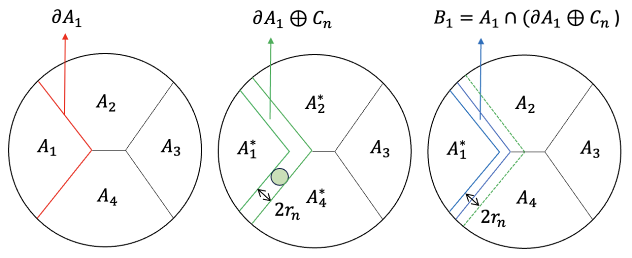

Fig. 8 Buffer region creation process via intersection of A_i sub-regions with Minkowski sum of boundary line and printer nozzle planar projection.

that all polygons in a Delaunay diagram are cyclic polygons (i.e., each triangle in a Delaunay triangulation has a unique circumcircle). In the general case, a Delaunay diagram may not have a unique triangulation. Consider a case where a quadrilateral cell can be split into two triangles in two different ways (by choosing to connect either of the diagonal vertices with an edge). However, when there are no more than three Voronoi sites that are cyclic, we always obtain a unique Delaunay triangulation [29]. This also means that the resulting Voronoi tessellation will always contain degree-3 vertices. This offers a principled approach to generating robot positions that ensures degree-3 partitioning. Therefore, we employ Voronoi tessellations to simultaneously control the geometry and the topology of the partition, as it naturally alleviates the key concerns in C3DP with robotic manipulators.

3.3 Buffer Regions. Creating a collision-free printing schedule based solely on the sub-regions A_i is too conservative. Collisions can generally only occur in the vicinity of shared boundaries of the partition, meaning robots can safely print in areas away from these interfaces. Hence, preventing two adjacent sub-regions from being printed in parallel due to a small (relative to A_i) shared boundary area is not ideal. This problem is intensified in C3DP applications due to the large size of the A_i sub-regions, as the ratio of the shared boundary area to the total area of A_i becomes smaller. To address this issue, we introduce the idea of *buffer regions* to improve the original printing schedule.

Recall that the collision-free schedule we derived for the symmetric partition of the circular layer (Fig 2(a)), has the same printing time as if only one printer was used. This is a clear example of how the printing schedule based solely on A_i can perform poorly due to its conservativeness. Therefore, we can improve the schedule, and thus the layer printing time for any partition, regardless of topology or geometry, by introducing buffer regions B_i. The idea is that the buffer regions will act as buffers between the original A_i sub-regions, such that they are the only regions of the layer geometry where collisions between robots may occur. This is guaranteed if the printer that was originally assigned to print sub-region A_i is also assigned to its corresponding buffer B_i, and the buffer region geometry is properly defined based on the printer dynamics. In general, determining the geometry of buffer regions requires knowledge of the mechanical model of the printers, that is, whether they are drones, mobile robots, or robotic arms [26]. However, regardless of the mechanical model, at a minimum one must consider the geometry of the nozzle when testing for collisions.

Let us represent the nozzle as a circle of radius r_n projected onto the layer surface. We assume, without loss of generality, that all printers have the same nozzle radius. Let the boundary of region A_i with all other regions A_j be ∂A_i. The buffer region B_i is then defined as the intersection of A_i with the Minkowski sum of ∂A_i and circle C_n of radius r_n (Fig. 8):

$$B_i = A_i \cap (C_n \oplus \partial A_i) \quad (6)$$

This process can be repeated for every A_i until all necessary buffer regions are created. Note that if ∂A_i is disjoint, so is B_i, although

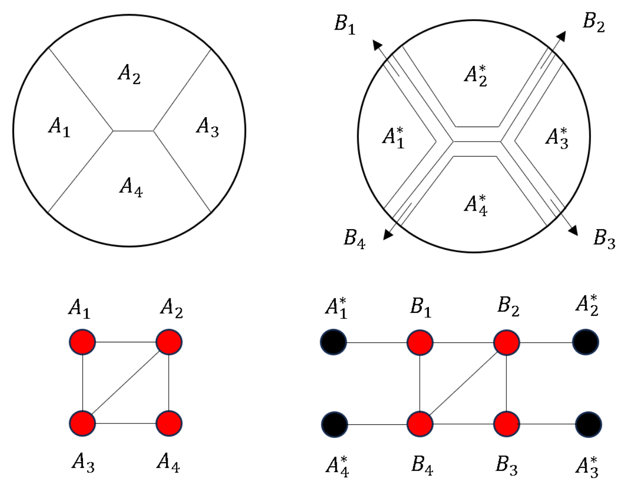

Fig. 9 The topology of the sub-graph representing the buffer region is the same as the original topology of the partition. This allows us to optimally print the buffer regions since it is guaranteed to be 3-colorable.

the buffer region is still treated as one singular node in the partition graph. Additionally, there is the possibility that B_i is empty, which can happen if the resulting intersection itself is empty. In this case, the node representing that particular B_i still exists in the partition graph, but it has an area value of zero.

A key property of the proposed buffer region creation process, is that if we analyze the topology of the resulting space, we note that 1) the new sub-regions A_i^*, formed by subtracting each B_i from their respective A_i, are only connected to their respective buffer regions B_i, which was our original goal, and 2) that the connectivity of the buffer regions between themselves is the same as the connectivity of A_i (Fig. 9). This result is clear when comparing the resulting adjacency matrix M_P^* of the new partition graph G_P^* with the old one:

$$M_P^* = \begin{bmatrix} 0 & I \\ I & M_P \end{bmatrix}, \tag{7}$$

where I is the identity matrix.

The algebraic form of the resulting adjacency matrix M_P^* (Eq. 7) is constant irrespective of the topology of the original partition graph. This result is important to SZ* as it allows the incorporation of an efficient heuristic-based scheduler to the current framework (Section 4.1.3) since the sub-graph formed by the B_i regions will always be 3-colorable, provided that the original partition graph is also 3-colorable. Therefore, once the buffer regions are efficiently printed using the 3-coloring schedule, all A_i^* can be safely printed in parallel during the final step of the schedule.

Consider that buffer regions are created for the symmetric partition of the circular layer (Fig. 2(a)). Then, the new schedule S^* becomes $((B_1), (B_2), (B_3), (B_4), \tau^*)$, where the tuple τ^* contains all A_i^*, yielding layer printing time T_l^*:

$$T_l^* = T(\tau^*) + \sum_{i=1}^{4} T(B_i), \tag{8}$$

which is clearly an improvement over the previous scenario since B_i are generally much smaller than A_i, especially in large-format printing which is what C3DP was designed for. For example, if the size of the area of the union of the buffer regions is a fraction c of the total layer geometry area, denoted here U, and we assume that the A_i^* sub-regions have roughly equal areas, then Eq. 8 can be rewritten as:

$$T_l^* = cT(U) + \frac{(1-c)}{4}T(U), \tag{9}$$

which converges to the ideal scenario (Eq. 1) as $c \to 0$, meaning that the buffer regions should ideally become a smaller proportion of the total layer geometry area U. Finally, creating these buffer regions using the proposed Minkowski sums can be applied to any partition, including user-defined ones.

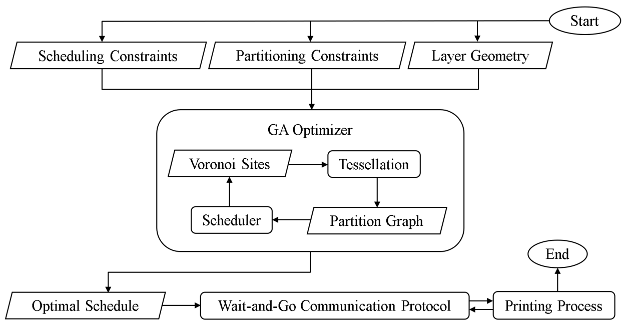

Fig. 10 Flowchart of the Safezone* methodology framework. Given the layer geometry and a set of constraints, the GA optimizer looks for Voronoi sites that minimize the makespan function calculated using the partition graph. The final schedule is sent to the control hub for printing.

4 Methodology

The overarching framework for SZ* consists of two core components:

(1) Co-Solving of Partitioning and Scheduling
(2) Printer Communication

The framework is modular in that it also allows for user-defined solutions. For example, if the user provides their own partitioning scheme by pre-defining the A_i sub-regions, SZ* is only responsible for creating the buffer regions (if desired), scheduling the printing, and overseeing the process through the established communication protocol using wait-and-go commands [26] (Fig. 10). However, in general, SZ* is responsible for co-solving partitioning and scheduling by co-considering layer geometry and topology in its design space exploration. The user can be interested in an user-defined partition for many reasons, but mainly because of the design of the part. For instance, if there is a multi-material process (PLA & ABS), then the partitions could be determined by the intended material deposition areas. The user might also want to avoid certain partitions that have boundaries in mechanically important regions, such as critical features that will be subjected to elevated stress and strain concentrations.

4.1 Space Partitioning and Scheduling.

The design space is defined to be the set of Voronoi sites $\{p_i\}$, which correspond to the printer locations (Section 3.2). The locations of the Voronoi sites directly influence the structure of geometry and topology at each layer through their associated Voronoi tessellation and resulting partition graph (Section 3.3). As discussed, we are interested in minimizing the makespan (T_l) of a layer, which can be extracted from the schedule of the partition graph. Thus, we can frame the choice of Voronoi sites as a constrained optimization problem, in which the objective function is the makespan function $C(p_i)$ of the partition graph. Before formally defining the optimization problem, we first describe the nature of the existing constraints.

4.1.1 Partitioning Constraints.

Within the context of C3DP, it is not uncommon for some robots to have constraints on physical locations and cannot be arbitrarily placed, limiting the placement of the Voronoi sites. As the location of the printers is directly related to the resulting partitioning, we call these constraints partitioning constraints. SZ* considers three such constraints when exploring the design space.

- *Reachability*: Given a radius $r_{p,i}$, defining a two-dimensional circular workspace $C_{p,i}$ centered at the Voronoi site $p_i \in \mathbb{R}$, and given a sub-region A_i, constructed as the intersection of the Voronoi cell of p_i and the layer geometry, then the reachability constraint states that A_i must be completely enclosed in $C_{p,i}$:

$$\forall i, \; A_i \subseteq C_{p,i} \tag{10}$$

  guaranteeing that the printers are able to reach their assigned sub-regions. Note that printers are also guaranteed to reach their buffer regions B_i, if they are created, since each B_i is enclosed within their A_i.

- *Separation*: Given a radius $r_{s,i}$, defining a circle $C_{s,i}$ centered at the Voronoi site $p_i \in \mathbb{R}$, then the separation constraint states that none of the circles $C_{s,i}$ should intersect with one another:

$$\forall i \neq j, \; C_{s,i} \cap C_{s,j} = \emptyset \tag{11}$$

  thus, we can tune the radii $r_{s,i}$ to create safe distances between each printer. This prevents printers to be placed too close to each other, which could result in collisions even before the printing process starts due to their spatial occupancy.

- *Configuration Space*: Given the configuration spaces $\mathcal{C}_i \in \mathbb{R}^2$ defining feasible values for the Voronoi sites p_i coordinates, and given their corresponding obstacle space $\mathcal{O}_i \in \mathbb{R}^2$ defined by the layer geometry, then the configuration space constraint enforces the following:

$$\forall i, \; p_i \in \mathcal{C}_i \setminus \mathcal{O}_i \tag{12}$$

  avoiding solutions in which the Voronoi sites p_i are inside of the layer geometry. If the layer geometry is not a genus-0 surface, meaning that it has inner holes, this constraint does not avoid the printer locations to be contained in those holes. However, if the user does not wish to consider those solutions, then the obstacle space $\mathcal{O}$ can be taken to be the union of the layer geometry and any associated holes.

Therefore, this set of partitioning constraints allows the SZ* framework to generate optimal solutions while maintaining feasibility in practical C3DP applications. Once the Voronoi sites are selected, the buffer regions B_i are created and the new partition graph G_P^* is constructed (Section 3.3). Finally, the objective function must be evaluated, which involves extracting the layer printing time (makespan) from the partition graph schedule.

4.1.2 Scheduling Constraints.

We can treat the nodes of G_P^* as tasks with time duration equal to the time taken for a printer to finish printing a sub-region (i.e., a task) that a node represents. As we are interested in deriving the optimal schedule S^* from the partition graph, which is in itself another optimization problem, we must first define how to evaluate the makespan function $C(p_i)$ and which physical constraints are involved.

Overall, SZ* considers two constraints in scheduling, which are derived from our previous layer topology analysis (Section 3.1), namely 1) that the printers cannot be working on two adjacent nodes at the same time, and 2) that each printer can only work on its associated safe region A_i^* and buffer region B_i. This is a standard graph scheduling problem that can be solved with a mixed-integer linear program (MILP). The problem is mixed-integer since there are continuous decision variables defining the starting time of the tasks, which can be any real positive number, and also binary decision variables defining the order in which the tasks are sequenced. Within the SZ* framework, we solve this inner optimization problem using the Computational Infrastructure for Operations Research Branch and Cut (COIN-OR CBC) solver [30].

Alternatively, we can leverage the analysis for the layer printing time T_l when adding buffer regions B_i (Section 3.3) and simplify the scheduling problem using a heuristic-based approach. The idea is to divide the scheduling problem into two smaller sub-problems. The first sub-problem refers to scheduling only the buffer regions B_i, while the second sub-problem involves scheduling the A_i^* sub-regions (Fig. 9). The second sub-problem is trivial as, by construction, all sub-regions A_i^* can be printed in parallel, and so no computational effort is needed. The first sub-problem can now be solved using the MILP optimizer, but considering only half the number of nodes when compared to the non-heuristic-based approach. Finally the schedules of the first and second sub-problems are appended, providing a new definition for $C(p_i)$. Although this approach does not guarantee an optimal solution, it is computationally efficient, and approximates the optimal makespan well when the size of the buffer region is small (Eq. 9).

4.1.3 Optimization with GA.

We are interested in finding a set $\{p_i\}$ of Voronoi sites that minimizes the makespan function $C(p_i)$ of the resulting partition graph. For notational simplicity, let the set of partitioning constraints (Section 4.1.1) be denoted C_1, and if a set $\{p_i\}$ of Voronoi sites satisfies all partitioning constraints, then $\{p_i\} \in C_1$. Thus, we can formalize the optimization problem as:

$$\begin{aligned} \min_{p_i} \quad & C(p_i) \\ \text{s.t.} \quad & \{p_i\} \in C_1 \end{aligned} \tag{13}$$

where the makespan function $C(p_i)$ can be user-defined.

As discussed in the previous section, the makespan function can be the optimal makespan of the partition graph, which could be found using any MILP solver, or it could be heuristic-driven, such that the buffer regions B_i are first printed using graph coloring, and then the safe regions A_i^* are printed in the final step. Irrespective, of how the makespan function $C(p_i)$ is defined, the formulation of (Eq. 13) remains the same.

To solve the optimization problem above, we employed a genetic algorithm (GA) formulation, where the gene is represented by the set of Voronoi site coordinates (which create the partition). The rationale for using a non-gradient-based optimizer is that the design space is not guaranteed to be continuous. Small perturbations in the location of the Voronoi sites may lead to partition graphs with distinct topologies creating discontinuities and a non-smooth makespan function. Therefore, a gradient-free, meta-heuristic optimizer such as the GA is an appropriate choice to solve (Eq. 13). To facilitate convergence, the partitioning constraints C_1 can be construed as penalty functions and added to C(p_i). For instance, the *reachability* constraint can be redefined as a penalty function $w(d)$, where d is the amount of area assigned to printer i that is outside of $C_{p,i}$, guaranteeing a smoother makespan function for better convergence. A similar rationale follows for the remaining partitioning constraints.

While other gradient-free and meta-heuristic optimizers are possible, such as simulated annealing (SA) or particle swarm optimization (PSO), we chose GA as it is more well-suited to high-dimensional, non-smooth design spaces such as ours. SA is a single-point optimizer that can have difficulties in exploring high-dimensional design spaces. Additionally, the determination of the cooling function is not trivial, complicating the criteria for convergence and increasing the chances of local-minima solutions. PSO was originally conceived for smoother problems, as the velocity-driven updates may struggle in rugged design spaces, leading to premature convergence. Hence, while a separate study can be conducted comparing these algorithms for our specific design problem, we defer it to future work.

4.2 Communication.

The last step in the SZ* framework is to establish the communication protocol during the printing process. The purpose of the protocol is to certify that the optimal schedule S^* is followed. We approach this problem from a centralized perspective, where a single control hub communicates independently with each printer and sends only two types of commands: *wait* or *go* [26]. The control hub follows the optimal schedule S^* generated through the partition graph G_P, which is a product of the Voronoi tessellation, to coordinate the sequence of *wait* and *go* commands. The steps are as follows:

(1) Iterate over each tuple of sub-regions τ in S^*. Stop if there are no more τ.
(2) Send the *go* command to each printer that has a region in τ.
(3) Once a printer finishes its region, send the *wait* command.
(4) Once all printers that have regions in τ receive the *wait* command, go back to Step 1.

Since the regions in each τ can be printed in parallel, this simple communication protocol guarantees a collision-free and optimal printing process. Note that SZ* is a framework that uses layer synchronization, meaning that the printers are not allowed to move on to the next layer until it is finished. This property is automatically enforced in our framework since each τ can only contain regions in the same layer. Layer synchronization is applied because errors between predicted and actual time to print can cause uncertainty (e.g., accumulated error) in planning that is not considered in the scheduling. This preserves the collision-free and layer-optimal scheduling properties outlined in the methodology.

5 Results

In this section we present a set of numerical studies using the proposed framework, as well as a physical validation demonstration using 3-DOF robotic manipulators arranged in a discrete grid. We demonstrate how SZ* is not only a valuable tool for practical C3DP applications, but also a powerful aid in exploring fundamental research questions regarding optimal partitioning and scheduling of 3D-printed parts.

5.1 Numerical Studies.

In order to demonstrate the viability of the SZ* framework, we conduct computational experiments using the proposed methodology and we postulate two hypotheses.

(1) Optimal Voronoi sites generate 3-colorable partition graphs.
(2) The areas of the A_i sub-regions will be approximately equal.

To show generality, we ran SZ* using a wide range of layer geometries, both convex and concave. Specifically, we tested four layer geometries: square, triangle, z-shape, and a cross section of the famous Stanford bunny. Additionally, we varied the number of printers n, and thus the number of Voronoi sites used, starting at $n = 4$ and going up to $n = 7$. The GA hyperparameters were kept constant at 50 generations, 100 population size, 20% mate rate, and 10% mutation rate. The partition constraint parameters were set at $r_s = 0.5$ and $r_p = 2.0$ units.

When generating results with this set of hyperparameters, we observed no issues in the convergence of the GA optimizer. However, due to the nature of the physical constraints, the user must be careful to ensure that the chosen physical scenario is realistic (e.g., adequate number of printers for the available space), or the GA may struggle to find a feasible solution. We have also provided the performance data for all numerical studies in Appendix A. Finally, all studies were performed on an HP Elite Tower 880 G9 computer with a 12th Gen Intel(R) Core(TM) i9-12900 2.40 GHz processor, with 64 GB of RAM.

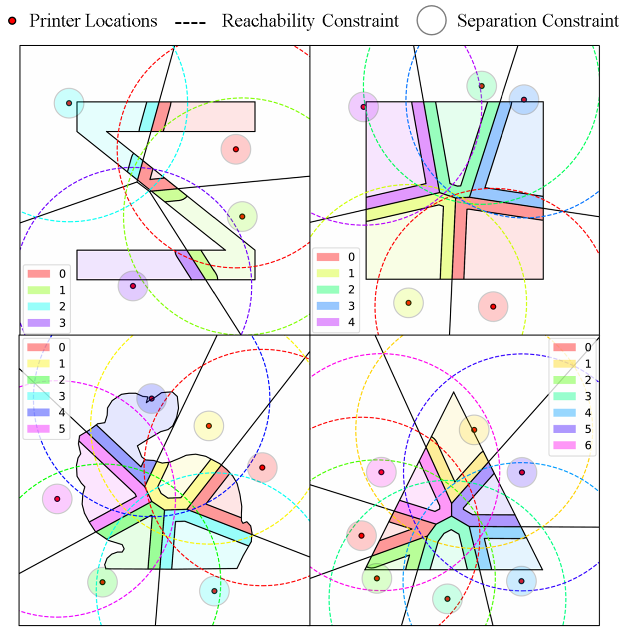

Fig. 11 Final printer positions using the SZ∗ framework. Starting from top left and going clockwise: z-shape with N=4, square with N=5, triangle with N=7, Stanford bunny with N=6.

5.1.1 *Numerical Results.*

For each combination of layer geometry and number of printers, and using the optimal MILP scheduler, we extracted the chromatic number of the resulting partition graph and analyzed the evolution of the areas of the sub-regions using a number of metrics.

For almost all of the scenarios, with one exception, the optimal solution had a partition graph with chromatic number $\chi(G) = 3$ (Fig. 11). In the scenario using the z-shaped printing part, with four printers, SZ* managed to find a solution with $\chi(G) = 2$, which is theoretically better, while satisfying all the constraints. This solution was possible due to the slender and concave nature of the printing part, allowing a path graph P_8 to be constructed. These results confirm our first hypothesis and the assertion made in the conceptual framework that, in general, topologies that are 3-colorable, or less if possible, are the best for C3DP.

To understand if there are underlying patterns in the layer geometry of these 3-colorable partitions, which would allow us to better guide the optimization process, we studied how the normalized areas of the sub-regions evolved with the generations. In particular, we looked at how the areas of the individual A_i^* and B_i sub-regions evolve, their sum, and their difference. For difference metrics, we looked at the difference between the maximum and the minimum areas, the standard deviation, the mean squared error, and the sum of the absolute differences. However, all metrics behaved similarly and no notable distinction was observed. For both the difference and the sum of the areas, there was no monotonic relationship to the makespan. One interesting pattern observed is that the individual areas themselves tend to form symmetric clusters. For example, in the case of $n = 4$, either all areas are equal, or there exists a 3-to-1 ratio, or a 2-to-2 ratio. Similar ratios were observed for higher values of n. Although the ratios change depending on the layer geometry, the areas were never observed to be randomly distributed (Fig. 12). The clustering pattern observed and the analysis of the areas is also seen for the other layer geometries considered.

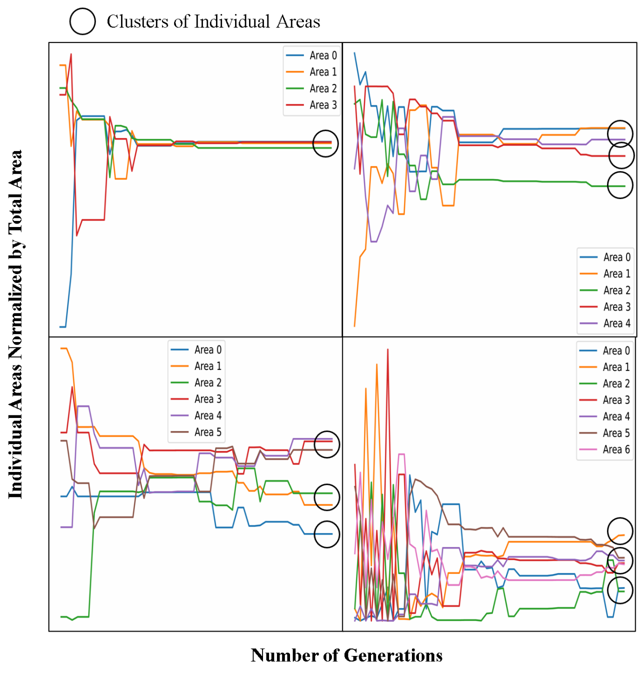

Fig. 12 Variation of the normalized A_i sub-regions as a function of the generations. The symmetric clustering of the areas is highlighted, showing different ratios for different geometries. Plots are a one-to-one match with the ordering in Fig. 12.

We posit that, in general, there is no simple partitioning geometry metric that relates to the makespan, implying that our original hypothesis regarding equality of areas is false. Therefore, minimizing the layer printing time does not necessarily yield a partition with sub-regions of equal area. Note that these conclusions are based on the partitioning strategy for SZ*, which uses $2n$ partitions and buffer regions, and may not necessarily hold for other families of partitions (Section 6.1).

5.2 Physical Validation. To physically validate SZ* we consider a scenario with four printers, where the part of interest is a solid disk of radius $100mm$. Since all the layers of the part are equal, and our framework uses layer synchronization, we only print one layer of the disk, which has a height of $0.45mm$. For printing, we used 4 SCARA printers developed by AMBOTS in a symmetric arrangement, sharing a square printing area with $600mm \times 600mm$ dimensions (Fig. 13). The choice of a disk, as opposed to a more complex geometry like the Stanford bunny cross-section, was made to simplify the visualization of layer partitioning and the printing sequence for demonstration purposes. However, our framework is fully capable of printing parts with complex geometries, as demonstrated by the numerical results.

The printers are equipped with a Duet controller board running the RepRap firmware. They are connected to a computer (control hub) through a local WiFi network. At the higher level, we have a Python script running on the control hub that oversees the printing process by following the wait-and-go communication protocol (Section 4.2). However, any programming language can be used as long as it supports sending HTTP requests. No external sensors besides the ones necessary for printer calibration, such as the BLTouch, were used.

5.2.1 Physical Results. In this manufacturing scenario, we ran SZ* with four printers to determine the optimal printer positions, and their corresponding partitioning and schedule, considering the disk layer geometry. However, the optimal printer positions found were not feasible, due to the grid-like nature of the experimental

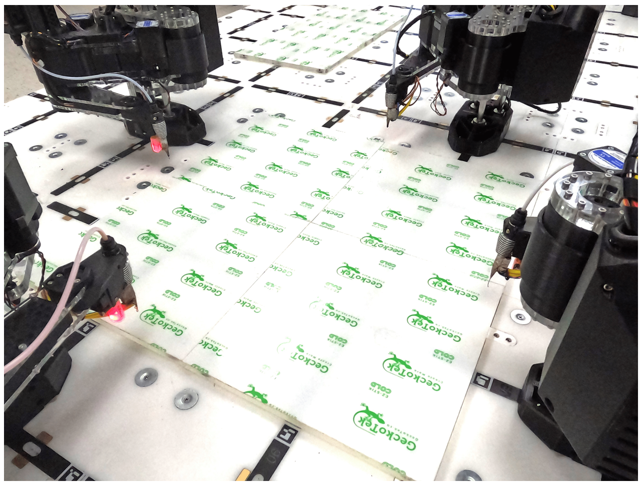

Fig. 13 Experimental setup for physical validation of SZ*. The printers can only be symmetrically arranged around the printer area as shown.

setup, forcing the printers to be placed in specific locations (Fig. 13). However, we observed that it was possible to use the tessellation given by SZ*, while keeping the printers in their feasible locations. This is because the feasible locations also satisfied the partitioning constraints (Section 4.1.1) for the optimal tessellation, allowing us to map the optimized printer positions to feasible ones without altering the partitioning or schedule.

Since the printer locations used in the optimization process differ from those used in actual printing, we call them *Virtual Robot Positions* (VRPs). These positions act as Voronoi sites in the optimizer but do not correspond to physical printer locations. Using VRPs is particularly beneficial in highly-constrained manufacturing scenarios, such as this one, as otherwise the design space would shrink significantly, restricting the types of partitions that SZ* can find. While the mapping between VRPs and feasible printer locations is trivial in this scenario, we recognize that this may not always be the case, warranting further exploration of this concept (Section 6.2).

For the purposes of physical validation, we partitioned the disk layer in two different ways. The first partition uses Voronoi sites that are co-located with the only feasible printer locations available, thus lying in a square, which is a cyclic polygon, producing a partition with a degree-4 vertex. When creating the second partition, we leverage the concept of VRPs and allow the GA optimizer to look for other solutions by decoupling the Voronoi sites from the printer locations. The resulting partition has a single degree-3 vertex, making it 3-colorable. (Fig. 14). The final printing schedule was computed using the heuristic-based approach in which the buffer regions B_i are printed first, followed by the safe regions A_i^*, and sent to the control hub for printing with the wait-and-go communication protocol. To facilitate visualization, we used different PLA filament colors for each printer: white, matte red, navy blue, and black. Therefore, regions A_1^* and B_1 are printed in white, regions A_2^* and B_2 in matte red, and so on (Fig. 15(a)-(d)). The time taken to print each sub-region and their respective areas is shown in Table 1. Time-lapses of both cases are also available[2].

Due to the presence of degree-4 nodes in the symmetric partition, we see that it lacks potential for parallelization when printing buffer regions, which ultimately impacts the layer printing time to be 2135 seconds or roughly 35 minutes. On the other hand, the degree-3 partition, which was created using our methodology, allows buffer regions B_2 (matte red) and B_4 (black) to be printed simultaneously, yielding a printing time of 1661 seconds

[2]Degree-3: https://tinyurl.com/4dwjsz28.
Symmetric: https://tinyurl.com/4subkebd

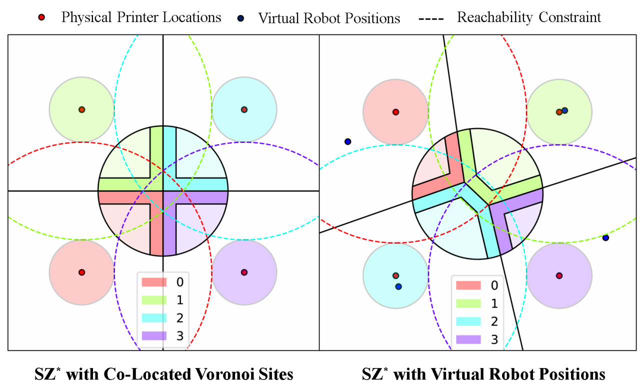
Physical Printer Locations
Virtual Robot Positions
Reachability Constraint
0
1
2
3
SZ* with Co-Located Voronoi Sites
SZ* with Virtual Robot Positions

Fig. 14 Leveraging the idea of virtual robot positions into the SZ* framework. An optimal and feasible partition is found although the printer locations were originally fixed by the manufacturing system.

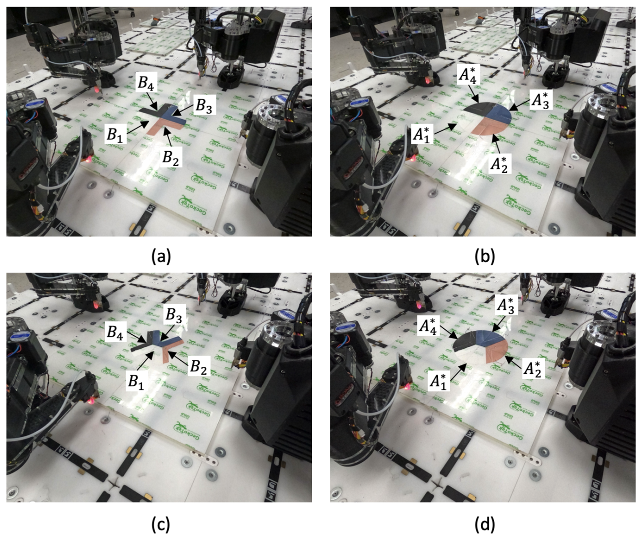

Fig. 15 Top: Degree-4 Symmetric Partition. Bottom: Degree-3 Partition with VRPs. (A & C) Buffer Regions. (B & D) Completed Print.

or approximately 28 minutes. Considering that the single-printer layer printing time T_s is about 3000 seconds (50 minutes), we see that the symmetric partition results in a printing time reduction of 28.83%, compared to 44.63% when using our degree-3 partition. Consequently, our degree-3 partition has a 22.20% reduction in printing time compared to the printing time of the symmetric case. It is important to note that the degree-3 partition has no reachability issues, which is enforced in our methodology by changing the form of the partitioning constraints with VRPs, thus making it collision-free, time-optimal, and applicable in real C3DP systems with robotic arms.

6 Discussion

SZ* is a promising framework for C3DP and demonstrates good performance in physical systems, however, there are several points that require further investigation. The proposed methodology currently limits the number of sub-regions created in the partition to be at most twice the number of printers. This limitation impedes SZ* to look for other families of partitions which may be of interest to C3DP. Additionally, a more general approach to using VRPs could be construed, as SZ* currently requires the printer positions to be fixed. We discuss some of these issues in the following subsections. One important topic that we do not address here is that of the mechanical properties of cooperatively-printed parts. However, some of our previous research has partially investigated this area for sloped-chunking strategies [11] and topologically-interlocked parts [24].

Table 1 Comparison of the two partitions generated in the physical validation. The superior degree-3 solution was only possible due to the concept of virtual robot positions. The time values are given in seconds, and the area values are given in millimeters squared.

Degree-4 Partition			Degree-3 Partition		
Region	Time	Area	Region	Time	Area
B_1	464	3848	B_1	487	5305
B_2	465	3848	B_3	473	5305
B_3	463	3848	B_2, B_4	345	7467
B_4	458	3848	-	-	-
A_i^*	285	16022	A_i^*	A_i^*	13337
TOTAL	2135	31414	TOTAL	1661	31414

6.1 Increasing Number of Partitions.

By design, SZ* partitions space into at most $2n$ sub-regions, where n is the number of printers used in the manufacturing process. This is always the case since the original n partitions, created by the Voronoi tessellation, can generate at most an additional n sub-regions during the buffer region creation process (Section 3.3). While it is possible to either modify SZ*, or devise new algorithms to consider other families of partitions, a number of challenges in the implementation must be considered. First, increasing the number of partitions also increases the complexity of the manufacturing process, as the printers have additional tasks that must be carefully coordinated. Second, more partitions increases the order of the partition graph G_P, which in turn increase the complexity of the graph scheduling problem. It is well-known that, due to the combinatorial nature of such graph scheduling problems, they do not scale well in the number of nodes. Finally, while our approach intelligently subdivides the original Voronoi cell into two, it is not trivial to devise a general partitioning and assigning rule that has the convenient properties of SZ^*, such as minimizing chances of collision and reducing issues with reachability. Challenges aside, we devote the remainder of this section to the analysis of interesting families of partitions, and their properties, which are of interest and may be explored in future research.

6.1.1 Ideal Partition Graphs. Consider, for example, a partition of the circular layer geometry using 3 printers that generates 9 sub-regions (Fig. 16(a)). The partition graph G_P for this topology is not only 3-colorable but also has order $|G_P|$ which is a multiple of its own chromatic number $\chi(G_P)$. Assuming that either all areas are equal, or all areas within each τ_j are equal, then the optimal schedule is $S = (\tau_1, \tau_2, \tau_3)$, where $\tau = (A_1, A_4, A_7)$, $\tau_2 = (A_2, A_5, A_8)$, and $\tau_3 = (A_3, A_6, A_9)$. This schedule yields the best possible layer printing time for a 3-printer scenario, being exactly a third of the single printer scenario. Hence, all such partition graphs, which we call ideal, should satisfy the following equality:

$$\chi(G_P) \cdot n = |G_P| \quad (14)$$

where $\chi(G_P)$ is the chromatic number of the partition graph, n is the number of printers, and $|G_P|$ is the order of the graph.

While using partitions that satisfy (Eq. 14) could be an interesting approach for C3DP systems, one must be careful to create graphs that minimize the chances of collision. As illustrated in the given ideal graph (Fig. 16(a)), the likelihood of collision is high and significant control effort would be required to guarantee safety, regardless of printer placement. Furthermore, without a kinematically redundant system, which is not guaranteed, it would be challenging to avoid collisions between the printers working on

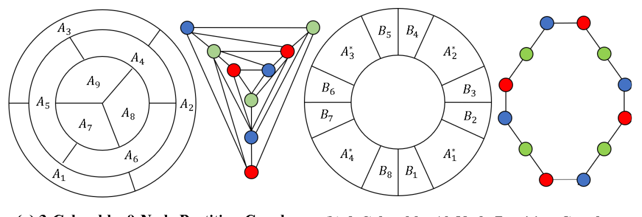
(a) 3-Colorable, 9-Node Partition Graph
(b) 3-Colorable, 12-Node Partition Graph

Fig. 16 Two partitions that yield the best possible printing time for 3 printers. Partition (A) may be created if we relax the configuration space constraint. Partition (B) may be created if disjoint buffer regions are treated as multiple regions.

A_1 and A_3 and the printer working on A_7. Hence, while this example may be well-suited for drones or mobile robots, it may lead to complications during implementation in systems with robotic manipulators. However, there may be other types of ideal graphs that have more favorable characteristics for robotic manipulators.

6.1.2 Partition Graphs in Higher-Genus Layers. Some layer geometries may have holes in them, that is, they may not be genus-0 surfaces. In such cases, if we consider disjoint sub-regions to be separate sub-regions, relaxing the assumption made in Section 4.2, then the underlying partition graph may change, potentially modifying its chromatic number, and allowing for a better printing schedule. Consider the case of a disk layer geometry, which is a genus-1 surface. Following the Voronoi tessellation, if we treat the buffer regions as separate we end up with 12 total sub-regions. In this case, buffer region B_1 becomes B_1 and B_2, buffer region B_2 becomes B_3 and B_4, and so on (Fig. 16(b)). This creates an even cycle graph C_{12} which is technically 2-colorable. However, following the rule for ideal graphs (Eq. 14), and given that the original partition considered four printers, we employ a 3-coloring scheme, which under the same area assumptions (Section 6.1.1), yields the best possible schedule. The key insight here is that such a partition would not have been possible if the layer geometry was genus-0, and we did not treat the original buffer regions as disjoint. Finally, while we presented two different partitioning strategies that may be interesting for C3DP applications, we do not discard that there may be others, as this enumeration was not exhaustive, but we leave this to future work.

6.2 Highly-Constrained Manufacturing Scenarios. As shown in the physical validation of SZ* (Section 5.2), it can be useful to decouple the Voronoi sites from the physical printer locations by leveraging the concept of VRPs in highly-constrained manufacturing scenarios. However, the mapping between the optimal printing positions and any set of feasible printing positions is not trivial. This is due to two main reasons. First, the mapping must ensure that all the partitioning constraints are satisfied, but those constraints were originally defined with respect to the VRPs, so additional work is needed. Second, given that this mapping is not a well-defined function, meaning that a single set of VRPs can map to multiple sets of feasible printer locations, we must explicitly define how the algorithm should behave. In this section we provide ideas for addressing these issues either assuming that there are user-defined fixed positions or not.

If we assume that the user determines the printer positions before the optimization process, then the mapping becomes easy, only leaving us with the constraint satisfaction problem. Since VRPs are not co-located with the printer positions, they have different partitioning constraints from the ones in the original framework (Section 4.1.1). Thus, if we have a user-defined set of fixed printer locations $\{q_i\}$, we can use the VRPs in the optimization framework by simply changing the *reachability* constraint to center the circular workspace $C_{q,i}$ around the fixed points $\{q_i\}$. In this case, it is expected that chosen locations already satisfy the *separation* and the *configuration space* constraints. With this new set of constraints, we can leverage the VRPs to search for a wider range of solutions (Fig. 17).

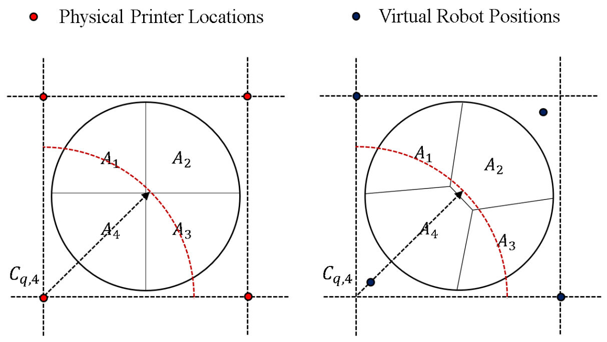

Fig. 17 Using virtual robot positions to find an optimal degree-3 partition while satisfying the reachability constraints of a grid-like manufacturing system.

On the other hand, if we assume that the user does not fix the location of the printers, then the mapping must be first defined. This is because the map transforming a set of VRPs to a feasible set of physical robot positions is not a well-defined function, that is, there could be multiple feasible printer locations associated with the optimal set of VRPs found. The simplest way to do this mapping is to choose the closest available printer location in the configuration space to that VRP. While this may work for the majority of cases, there can be scenarios in which doing so would result in a violation of the separation constraint, meaning that two printers are placed too closed to each other. Thus, we see that there is no simple rule that can define this mapping while guaranteeing a feasible set of physical locations. In fact, this mapping can be formally framed as a constraint satisfaction problem (CSP), in which we are interested in finding a feasible set of physical printer locations. In the framework of the CSP, the set of variables is the physical printer locations, the domain is given by the workspace of the manufacturing system, and the set of constraints is the partitioning constraints. While the CSP formulation can be efficiently solved using state-of-the-art algorithms, it would induce a higher computational burden in the SZ* framework. Therefore, a more detailed study regarding when VRPs should be used, given the level of freedom in printer positioning in a specific manufacturing system, is required.

6.3 Considerations for Printer Positioning in Multi-Layer Optimization. SZ* is inherently a layer-by-layer framework, as layer geometry may vary between layers. Consequently, for a 3D-printed part with changing layer geometry, each printer may have a different optimal location depending on the layer number. While this poses no algorithmic challenge to SZ*, as the current framework assumes that the printers are mobile and can be repositioned before starting each new layer, we recognize that frequent movement may affect printer calibration procedures, which some users may wish to avoid. Therefore, this section explores how SZ* can be adapted to minimize inter-layer movement while preserving its optimal properties.

A key observation is that, for most 3D-printed parts, the layer-by-layer transitions are relatively smooth, meaning discontinuous changes in layer geometry are uncommon. Consequently, the optimal tessellation is likely to change only slightly between layers, resulting in minimal adjustments to printer positions. By leveraging the concept of VRPs (Section 6.2) and decoupling the Voronoi sites from printer positions, the same printer positions can be maintained across multiple optimal tessellations, reducing the need for repositioning between layers. Between layers where discontinuities occur, movement may be necessary. However, SZ* can be adapted to reuse printer positions as much as possible, ensuring that they satisfy reachability constraints for each optimal tessellation.

If printer movement between layers is not feasible, SZ* would need to be adapted to perform tessellation at the 3D level. This can be achieved by retaining the existing optimization framework while modifying the evaluation function to assess the schedule across all layers for a single tessellation. In this approach, an estimate of the total print makespan would guide tessellation decisions. However, this method would likely yield more suboptimal solutions, compared to the layer-by-layer approach, due to the lack of flexibility in printer repositioning. Additionally, it would be computationally expensive, as each solution would need to be evaluated across all layers of the printed part. Therefore, it is not trivial to design an efficient algorithm to find optimal tessellation at the 3D level for C3DP systems.

7 Conclusion

In this paper, we developed and physically validated SafeZone*, a collision-free and time-optimal C3DP framework. The results show that the framework is viable, scalable, and has a lower printing time than traditional C3DP methods available in the literature. The key conceptual contribution of our algorithm is the co-solving of the partitioning and scheduling problems through an integrated geometric-topological analysis. However, based on the discussion topics presented, there are many interesting research directions that could further enhance our understanding of C3DP and the theory on which SafeZone* is grounded.

Acknowledgment

This work was partially supported by the National Science Foundation (NSF) Award #2048182 (Engineering Design and Systems Engineering Program) and the Army Research Lab (ARL) via grant number W911NF-24-2-0007. 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 or the ARL.

Appendix A: Performance Data for GA Optimization of Numerical Cases

In this appendix we present the performance data for the numerical studies presented in the paper. We consider two key parameters: algorithm runtime and a sub-optimality factor. The algorithm runtime is determined by how long the optimizer takes, in seconds, to converge to the 50th generation using our choice of hyperparameters (Section 5.1.1). The sub-optimality factor is defined as:

$$SO = \frac{C_{sol}}{C_{opt}} \tag{A1}$$

where C_{sol} is the makespan of the final solution found at the 50th generation, and C_{opt} is the theoretical best makespan, which is given by dividing the time to print the entire layer geometry using one printer by the number of printers used. The optimal solution has a sub-optimality factor of 1, and non-optimal solution will have a sub-optimality factor greater than 1. Note that the sub-optimality factor is conservative in the sense that there is no guarantee that any partitioning and scheduling algorithm that uses at most $2n$ subregions can achieve the theoretical best solution. The results are summarized in Table 2.

Additionally, we have also shown one example for the convergence graph of the fitness of our GA optimizer (Fig. 18). A similar pattern was observed for all numerical results, regardless of geometry and number of printers. Note that the theoretical best show in the figure is coincidentally equal to one, as the area of the z-beam is 4 units and there are 4 printers. The signs are flipped due to the internal implementation attempting to maximize the fitness function.

Table 2 Performance data for the numerical studies. We assess the algorithm runtime and the sub-optimality factor.

Cross-Section	No. Printers	Time (s)	Sub-Optimality Factor
Bunny	4	140.299	1.089
Bunny	5	110.427	1.179
Bunny	6	334.585	1.343
Bunny	7	546.572	1.530
Square	4	12.557	1.146
Square	5	62.083	1.172
Square	6	210.670	1.328
Square	7	261.470	1.539
Triangle	4	130.301	1.076
Triangle	5	301.831	1.290
Triangle	6	355.738	1.643
Triangle	7	603.639	1.917
Z-Beam	4	107.233	1.009
Z-Beam	5	69.659	1.066
Z-Beam	6	113.868	1.083
Z-Beam	7	210.876	1.059

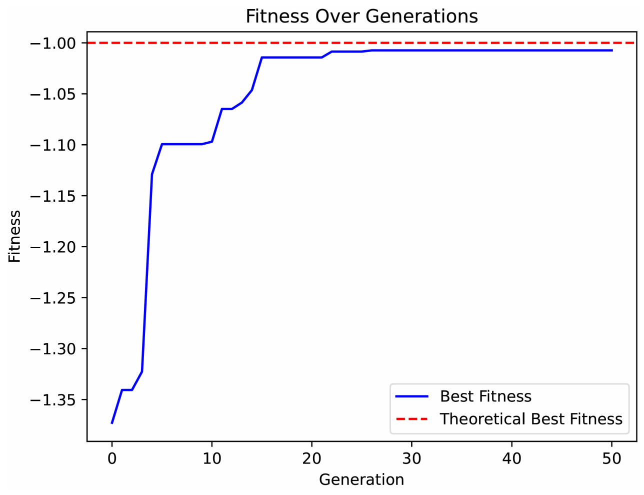

Fig. 18 Sample convergence graph for GA optimization, showing the fitness (makespan) of the z-beam geometry with 4 printers. A similar pattern is observed in the other numerical studies.

References

[1] Abdullah Alhijaily, Z. M. K. and Bartolo, A. N. P., 2023, "Teams of robots in additive manufacturing: a review," Virtual and Physical Prototyping, **18**(1), p. e2162929.

[2] Rescsanski, S., Hebert, R., Haghighi, A., Tang, J., and Imani, F., 2025, "Towards intelligent cooperative robotics in additive manufacturing: Past, present, and future," Robotics and Computer-Integrated Manufacturing, **93**, p. 102925.

[3] Poudel, L., Marques, L. G., Williams, R. A., Hyden, Z., Guerra, P., Fowler, O. L., Sha, Z., and Zhou, W., 2022, "Toward Swarm Manufacturing: Architecting a Cooperative 3D Printing System," Journal of Manufacturing Science and Engineering, **144**(8), p. 081004.

[4] Zhang, K., Chermprayong, P., Xiao, F., Tzoumanikas, D., Dams, B., Kay, S., Kocer, B. B., Burns, A., Orr, L., Choi, C., et al., 2022, "Aerial additive manufacturing with multiple autonomous robots," Nature, **609**(7928), pp. 709–717.

[5] Xu, X., Wang, Z., and Feng, C., 2021, "Projector-Guided Non-Holonomic Mobile 3D Printing," 2105.08950

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

[7] Khosravani, M. R. and Haghighi, A., 2022, "Large-scale automated additive construction: overview, robotic solutions, sustainability, and future prospect," Sustainability, **14**(15), p. 9782.

[8] Stone, R. F., Ebert, M., Zhou, W., Akleman, E., Krishnamurthy, V., and Sha, Z., 2024, "SafeZone: A Topologically-Aware Voronoi-Based Framework

for Fast Collision-Free Cooperative 3d Printing," *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 88346, American Society of Mechanical Engineers, p. V02AT02A038.

[9] Gao, W., Zhang, Y., Ramanujan, D., Ramani, K., Chen, Y., Williams, C. B., Wang, C. C., Shin, Y. C., Zhang, S., and Zavattieri, P. D., 2015, "The status, challenges, and future of additive manufacturing in engineering," Computer-Aided Design, **69**, pp. 65–89.

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

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

[12] Larsen, L., Pham, V.-L., Kim, J., and Kupke, M., 2015, "Collision-free path planning of industrial cooperating robots for aircraft fuselage production," 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 2042–2047.

[13] Fei, Y., Fuqiang, D., and Xifang, Z., 2004, "Collision-free motion planning of dual-arm reconfigurable robots," Robotics and Computer-Integrated Manufacturing, **20**(4), pp. 351–357.

[14] Lin, H.-Y. and Huang, Y.-C., 2021, "Collaborative complete coverage and path planning for multi-robot exploration," Sensors, **21**(11), p. 3709.

[15] Gul, F., Mir, A., Mir, I., Mir, S., Islaam, T. U., Abualigah, L., and Forestiero, A., 2022, "A Centralized Strategy for Multi-Agent Exploration," IEEE Access, **10**, pp. 126871–126884.

[16] Karapetyan, N., Benson, K., McKinney, C., Taslakian, P., and Rekleitis, I., 2017, "Efficient multi-robot coverage of a known environment," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 1846–1852.

[17] Zbiss, K., Kacem, A., Santillo, M., and Mohammadi, A., 2022, "Automatic collision-free trajectory generation for collaborative robotic car-painting," IEEE Access, **10**, pp. 9950–9959.

[18] Alhijaily, A., Kilic, Z. M., and Bartolo, P., 2023, "Online cooperative printing by mobile robots," Virtual and Physical Prototyping, **18**(1), p. e2276257.

[19] Alhijaily, A., Kilic, Z. M., and Bartolo, A. N. P., 2023, "Teams of robots in additive manufacturing: a review," Virtual and Physical Prototyping, **18**(1), p. e2162929.

[20] Poudel, L., Zhou, W., and Sha, Z., 2021, "Resource-Constrained Scheduling for Multi-Robot Cooperative Three-Dimensional Printing," Journal of Mechanical Design, **143**(7), 072002.

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

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

[23] Poudel, L., Zhou, W., and Sha, Z., 2020, "A generative approach for scheduling multi-robot cooperative three-dimensional printing," Journal of Computing and Information Science in Engineering, **20**(6), p. 061011.

[24] Ebert, M., Stone, R., Koithan, J., Zhou, W., Pharr, M., Estrin, Y., Akleman, E., Sha, Z., and Krishnamurthy, V., 2025, "NoodlePrint: Cooperative Multi-Robot Additive Manufacturing with Helically Interlocked Tiles," Journal of Manufacturing Science and Engineering, pp. 1–21.

[25] Krishnamurthy, V., Poudel, L., Ebert, M., Weber, D. H., Wu, R., Zhou, W., Akleman, E., and Sha, Z., 2022, "LayerLock: Layer-Wise Collision-Free Multi-Robot Additive Manufacturing Using Topologically Interlocked Space-Filling Shapes," Computer-Aided Design, **152**, p. 103392.

[26] Stone, R. F., Zhou, W., Akleman, E., Krishnamurthy, V. R., and Sha, Z., 2023, "Print As a Dance Duet: Communication Strategies for Collision-Free Arm-Arm Coordination in Cooperative 3D Printing," *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 87295, American Society of Mechanical Engineers, p. V002T02A081.

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

[28] Jensen, T. R. and Toft, B., 2011, *Graph coloring problems*, John Wiley & Sons.

[29] Okabe, A., Boots, B., Sugihara, K., and Chiu, S. N., 2009, *Spatial tessellations: concepts and applications of Voronoi diagrams*, John Wiley & Sons.

[30] Saltzman, M. J., 2002, "COIN-OR: an open-source library for optimization," Programming languages and systems in computational economics and finance, pp. 3–32.

List of Figures

List of Tables