Robotics: Science and Systems 2020
Corvalis, Oregon, USA, July 12-16, 2020

Scalable and Probabilistically Complete Planning for Robotic Spatial Extrusion

Caelan Reed Garrett[†*], Yijiang Huang[‡*], Tomás Lozano-Pérez[†] and Caitlin Tobin Mueller[‡]
[†]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology
Email: {caelan,tlp}@csail.mit.edu
[‡]Department of Architecture, Massachusetts Institute of Technology
Email: {yijiangh,caitlinm}@mit.edu
[*]Authors contributed equally

***Abstract*—There is increasing demand for automated systems that can fabricate 3D structures. Robotic spatial extrusion has become an attractive alternative to traditional layer-based 3D printing due to a manipulator's flexibility to print large, directionally-dependent structures. However, existing extrusion planning algorithms require a substantial amount of human input, do not scale to large instances, and lack theoretical guarantees. In this work, we present a rigorous formalization of robotic spatial extrusion planning and provide several efficient and probabilistically complete planning algorithms. The key planning challenge is, throughout the printing process, satisfying both *stiffness* constraints that limit the deformation of the structure and *geometric* constraints that ensure the robot does not collide with the structure. We show that, although these constraints often conflict with each other, a greedy backward state-space search guided by a stiffness-aware heuristic is able to successfully balance both constraints. We empirically compare our methods on a benchmark of over 40 simulated extrusion problems. Finally, we apply our approach to 3 real-world extrusion problems.**

I. Introduction

Spatial frame structures are used extensively in architecture to represent objects that cannot be easily captured by surfaces or volumetric solids (*e.g.* the Klein bottle in Figure 1). These structures are useful due to their high strength-to-weight ratios [44, 21]. Extrusion-based methods, such as 3D printing, can effectively fabricate these geometrically and topologically complex structures. Most existing printing systems deploy a 2.5D strategy where melted materials are accumulated layer upon layer along a fixed direction. These systems are unable to print general 3D frame structures due to their inability to print in arbitrary directions. Robot manipulators have proven to be viable alternatives for fabricating these structures due to their additional capabilities afforded by extra degrees-of-freedom (DOFs) [14, 49, 44, 37]. However, robotic spatial extrusion has only been applied in limited capacities due to the planning challenges imposed when fabricating large, irregular structures. The robot must respect both collision and kinematic *geometric* constraints present in manipulation tasks, and each partial-structure must respect *structural* constraints that ensure feasible construction. In extrusion planning, a *stiffness* constraint, which prevents significant structural deformation, is the primary structural constraint. Existing algorithms both require strong *human* guidance to solve these problems [22] and lack completeness guarantees [47, 20].

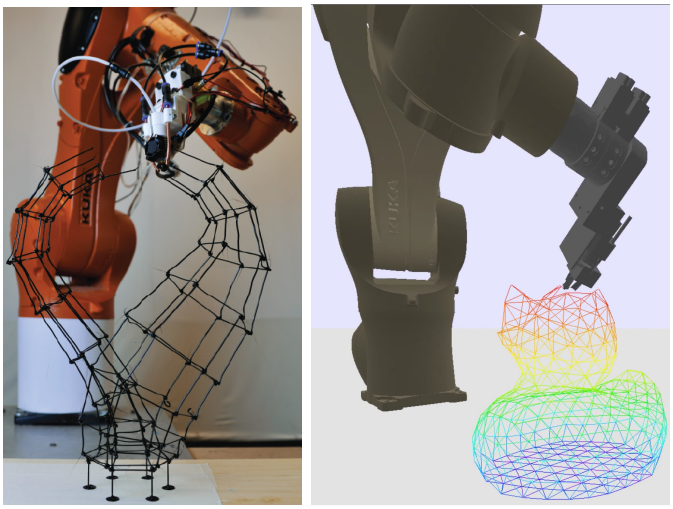

Fig. 1. *Left*: Real-world Klein bottle extrusion (246 elements). *Right*: Simulated duck extrusion (909 elements).

We present an algorithmic treatment of extrusion planning that focuses on its mathematical form, probabilistically complete algorithms, and algorithms that scale empirically. We identify a dichotomy between satisfying geometric and structural constraints; stiffness most significantly impacts decisions at the *beginning* of construction while collisions most significantly limit actions towards the *end* of construction. In isolation, forward search is most effective for stiffness constraints but backward search is most effective for geometric constraints. We provide algorithms that efficiently plan in the presence of both constraints by globally performing a greedy backward search, using forward reasoning to bias the search towards stiff structures. The contributions of this paper are:

1) A formalization of robotic spatial extrusion in the presence of stiffness and geometric constraints;
2) Efficient and probabilistically complete forward and backward state-space search algorithms;
3) Prioritization heuristics that guide both stiffness and geometric decision-making;
4) An investigation of the failure cases of these methods;
5) Validation of our methods both on long-horizon simulated and real-world extrusion problems.

II. Related Work

Most existing work on extrusion planning only addresses planning for a free-flying end effector. Wu *et al.* gave an algorithm for planning without stiffness constraints that considers a fixed discretization of end-effector orientations. It performs backward peeling [47] and computes a partial-ordering of elements that respects collision constraints. Then, it orders elements in a manner that preserves connectivity and the partial ordering. However, this procedure is incomplete because it rigidly commits to a single partial ordering. Huang *et al.* proposed a constrained graph decomposition algorithm to guide the extrusion sequence search [20]; however, their algorithm is also incomplete. Gelber *et al.* presented a complete forward search algorithm for a 3-axis printer that minimizes the deformation of a structure [11]. Choreo is the first extrusion planning system using a robot manipulator [22]. Choreo decomposes extrusion planning into a *sequence planning* phase, where it plans each extrusion, and a *transit planning* phase, where it plans motions between each extrusion. Because of this strict hierarchy, Choreo is incomplete as it is unable to backtrack in the event that transit planning fails to find a motion plan. Choreo performs a forward search during sequence planning, using constraint propagation to prune unsafe end-effector orientations. To make sequence planning tractable, Choreo requires a user-generated partial ordering on elements.

Task and Motion Planning (TAMP) involves planning both the high-level objectives as well as the low-level robot motions required to complete a multi-step manipulation task [42, 45, 10]. For extrusion planning, the high-level decisions are the extrusion sequence, and the low-level motions are the extrusion and transit trajectories of the robot. A key challenge of extrusion planning when compared to typical TAMP problems is that its planning horizon is often substantially longer. Solutions to most TAMP benchmarks involves fewer than 50 high-level actions [30], while extrusion problems may require over 900 extrusions (Figure 1). At the same time, extrusion planning is less general than TAMP in several ways: 1) there is a single goal state 2) the robot's configuration is the only continuous state variable 3) every solution is an alternating sequence of movements and extrusions of a known length. Similar to how specializing to pick-and-place subclasses of TAMP enables the design of efficient algorithms [28, 15], we take advantage of these restrictions and structural properties to develop efficient algorithms that scale to large problems.

Extrusion planning can framed as *Multi-Modal Motion Planning* (MMMP) [17, 18], motion planning subject to a sequence of *mode* constraints σ on the feasible configuration space of the robot $\mathcal{M}(\sigma) \subseteq \mathcal{Q}$. Often times, $\mathcal{M}(\sigma)$ might be a lower-dimensional submanifold of an ambient space $\mathcal{Q}$. A critical component of MMMP is identifying *transition configurations* $q \in \mathcal{T}(\sigma, \sigma') \subseteq (\mathcal{M}(\sigma) \cap \mathcal{M}(\sigma'))$ between modes σ, σ', which allow for a discrete *mode switch* from $\sigma \to \sigma'$. Hauser and Ng-Thow-Hing provide an algorithm for MMMP that performs a forward state-space search through the space of modes [18]. They prove that their algorithm is *probabilistically complete* [24, 32], namely that it will solve any *robustly feasible* [23] MMMP problem with probability one. However, their algorithm blindly explores the state-space, which is intractable for the problems we consider.

III. Extrusion Sequencing

We begin by formulating spatial extrusion planning in the *absence* of a robot. A *frame* structure is an *undirected geometric graph* $\langle N, E \rangle$ embedded within $\mathbb{R}^3$. Let the graph's vertices N be called *nodes* and the graph's edges be called *elements* $E \subseteq N^2$ where $m = |E|$. Each node $n \in N$ is the connection point for one or more elements at position $p_n \in \mathbb{R}^3$. Each element $e = \{n, n'\} \in E$ occupies a volume within $\mathbb{R}^3$ corresponding to a cylinder of revolution about the straight line segment $p_n \to p_{n'}$. A subset of the nodes $G \subseteq N$ are rigidly fixed to *ground* and thus experience a reaction force. Each element $e = \{n, n'\}$ can either be extruded from $n \to n'$ or $n' \to n$. Let *directed* element $\vec{e} = \langle n, n' \rangle$ denote extruding element $e = \{n, n'\}$ from $n \to n'$. We will use the set $P \subseteq E$ to refer to a set of printed elements, representing a partially-extruded structure. Let $N_P = G \cup \{n, n' \mid \{n, n'\} \in P\} \subseteq N$ be the set of nodes spanned by ground nodes G and elements P. Extrusion planning requires first finding an *extrusion sequence*, an ordering of directed elements $\vec{\psi} = [\vec{e}_1, ..., \vec{e}_m]$. We will use ψ to denote the undirected version of $\vec{\psi}$. Let $\vec{\psi}_{1:i} = [\vec{e}_1, ..., \vec{e}_i]$ give the first i elements of $\vec{\psi}$ where $i \leq m$.

A. Stiffness Constraint

The key structural invariant that must hold throughout the extrusion process is a *stiffness constraint* requiring the maximal nodal deformation to be below a given tolerance. Each element experiences a self-weight load due to gravity, which causes the structure to bend. If the displacement is too large, elements might not successfully connect at the intended nodes. We approximate uniformly-distributed self-weight loads by applying half the load at each end of the element and using the fixed-end beam equation for moment approximation [12]. The deformation of all the nodes is calculated using finite element analysis of linear frame structures [35]. For a 3D frame structure, each node has six degrees of freedom (DOF) $(u_x, u_y, u_z, \theta_x, \theta_y, \theta_z)$, which correspond to the translational and rotational nodal displacements in the global coordinate system. Using linear basis functions and the local-to-global frame transformation, we can derive the beam equation to link the nodal load to nodal displacement in the *global* coordinate system [35] $K_e \left(\mathbf{u}_n, \mathbf{u}_{n'}\right)^T = \mathbf{f}_e$. Then, by concatenating all nodal DOF into a vector $\mathbf{u} = (..., u_{x,n}, u_{y,n}, u_{z,n}, \theta_{x,n}, \theta_{y,n}, \theta_{z,n}, ...)$ for $n \in N$, the system stiffness matrix K is assembled using:

$$K_{ij} = \begin{cases} \sum_{e \sim (i,j)} K_e(\text{e-dof}(i), \text{e-dof}(j)) & \text{if } i \sim j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $i \sim j$ indicates that the nodal DOFs $i, j \in \{1, ..., 6|N|\}$ are connected by an element, $e \sim (i, j)$ indicates that element e connects DOFs i, j, and e-dof(i) gives the corresponding

index of the DOF i in the local element system. The support condition specifies a set of fixed nodal DOF indices $\{s_1, \cdots, s_{6|G|}\} \subset \{1, \cdots, 6|N|\}$. The assembled system stiffness equation $K\mathbf{u} = \mathbf{F}$ is rearranged in the form:

$$\begin{pmatrix} K_{ff} & K_{fs} \\ K_{sf} & K_{ss} \end{pmatrix} \begin{pmatrix} \mathbf{u}_f \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{F_f} \\ \mathbf{F_s} \end{pmatrix} \tag{2}$$

The submatrix K_{ff} is positive definite (PD) if all elements are transitively connected to a ground node. Then, the nodal displacement under the structure's load can be obtained by solving the following sparse PD linear system: $K_{ff}\mathbf{u}_f = \mathbf{F}_f$. Let the procedure STIFF(G, P) test whether a partially-extruded structure P with ground nodes G satisfies the given maximum displacement tolerance.

Definition 1. An extrusion sequence $\vec{\psi} = [\vec{e}_1, \vec{e}_2, ..., \vec{e}_m]$ is *valid* if $\{e \in \psi\} = E$ and $\forall i \in \{1, ..., m\}$. STIFF$(G, \vec{\psi}_{1:i})$ and $n_i \in N_{\vec{\psi}_{1:i-1}}$ where $\vec{\psi}_i = \vec{e}_i = \langle n_i, n_i' \rangle$.

IV. Robotic Extrusion

We consider extrusion planning performed by a single articulated robot manipulator with d DOFs. Let $\mathcal{Q} \subset \mathbb{R}^d$ be the bounded configuration space of the robot where $q \in \mathcal{Q}$ is a robot configuration. The robot executes continuous trajectories $\tau : [0, 1] \to \mathcal{Q}$ where $\tau(\lambda) \in \mathcal{Q}$ is the robot's configuration at time λ for $\lambda \in [0, 1]$. The robot must adhere to its joint limits as well as avoid collisions with itself, the environment, and the currently printed elements. Let $Q : P \to \mathcal{Q}$ be a function that maps a set of printed elements $P \subseteq E$ to the collision-free configuration space of the robot $Q(P) \subseteq \mathcal{Q}$. When no elements have been printed, $Q(\emptyset)$ is the collision-free configuration space of the robot when only considering environment collisions, self-collisions, and joint limits. Each additionally printed element weakly decreases the collision-free configuration space, *i.e.*

$$P \subseteq P' \implies Q(P') \subseteq Q(P). \tag{3}$$

To ensure τ can be safely executed given printed elements P, $\forall \lambda \in [0, 1]$. $\tau(\lambda) \in Q(P)$. Finally, let $f_p(q) = x_p \in \mathbb{R}^3$ and $f_o(q) = x_o \in \mathrm{SO}(3)$ be the forward kinematic equations for the position and orientation of the end effector when the robot is at configuration q.

A. Extrusion

The robot extrudes material at the position of its end effector while executing an *extrusion trajectory* τ_e, which prints the continuous curve $l(\lambda) = f_p(\tau(\lambda))$. Thus, element $\vec{e} = \langle n, n' \rangle$ can be extruded by following a trajectory $\tau_{\vec{e}}$ if $\forall \lambda \in [0, 1]$:

$$||\lambda p_n + (1 - \lambda)p_{n'} - f_p(\tau_e(\lambda))|| = 0. \tag{4}$$

To prevent the end effector from colliding with the element while it is being extruded, the orientation of the end effector x_o is constrained be within the hemisphere $X_o(\vec{e})$, the set of orientations opposite to the direction of $p_n \to p_{n'}$:

$$X_o(\langle n, n' \rangle) = \{x_o \in \mathrm{SO}(3) \mid (p_{n'} - p_n)^\intercal (x_o \cdot [0, 0, 1]^\intercal) \leq 0\}.$$

Additionally, we enforce that the end-effector orientation x_o remains constant while extruding the element, $\forall \lambda \in [0, 1]$, $||x_o - f_o(\tau(\lambda))|| = 0$ to prevent the extruded material from inducing a twisting force. In practice, we also require the robot to perform *retraction* motions that move into and out of contact with the extruded element without extruding any material. Let $\rho \geq 0$ be an end-effector retraction distance hyperparameter. Then, the retraction position for node n at end-effector orientation x_o is: $r(n, x_o) = p_n + (x_o \cdot [0, 0, -\rho]^\intercal)$. Thus, the end effector moves from $r(n, x_o) \to p_n$ before extruding $\vec{e}$ and from $p_{n'} \to r(n', x_o)$ after extruding $\vec{e}$. We will treat retraction as a component of an extrusion motion. See Figure 2 for a visualization of each motion type.

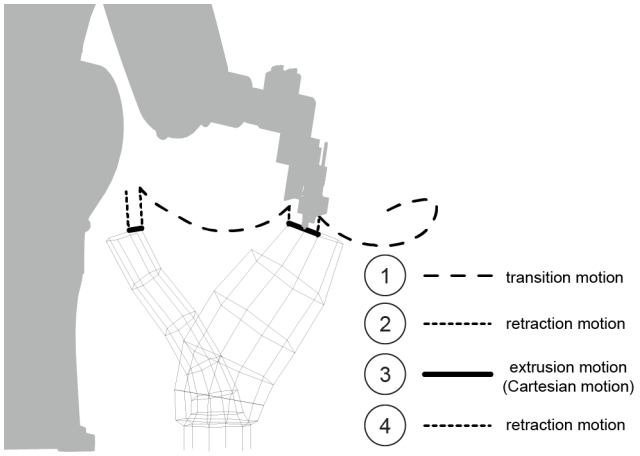

Fig. 2. Transition, retraction, and extrusion motions for two elements.

B. MMMP Formulation

Viewing extrusion planning under this lens of MMMP is valuable for understanding the geometry of the problem and its impact on completeness. Extrusion planning has two *mode families*, parameterized mode forms. A single *transit mode* (denoted as α) governs the robot's movement while *not* extruding [1, 41]. The only active constraint is trivially that $q \in \mathcal{Q}$. Any probabilistically complete motion planner PLANMOTION, such as a Rapidly-Exploring Random Tree (RRT) [31, 26], can be used to plan within transit modes.

An *extrusion mode* $\sigma_{\vec{e}} = x_o \in X_o(\vec{e})$ governs the robot's motion while extruding element $\vec{e} = \langle n, n' \rangle$ by starting at point p_n and ending at $p_{n'}$. Here, x_o is a continuous *coparameter* that defines the end-effector orientation constraint. Because of the position and orientation constraints on the end-effector, $\mathcal{M}(\sigma_{\vec{e}}) \subset \mathcal{Q}$ is a $(d-5)$-dimensional submanifold of the ambient space $\mathcal{Q}$. As typical in constrained motion planning, we enforce that any trajectory τ operating subject to mode σ stays within an ϵ-neighborhood of $\mathcal{M}(\sigma)$ [43]. Let $\delta(q, \mathcal{M}(\sigma)) = \inf_{q' \in \mathcal{M}(\sigma)} ||q - q'||$ be minimum distance from configuration q to $\mathcal{M}(\sigma)$ and $\gamma(\tau, \mathcal{M}(\sigma)) = \sup_{\lambda \in [0,1]} \delta(\tau(\lambda), \mathcal{M}(\sigma))$ be the maximum distance from trajectory τ to $\mathcal{M}(\sigma)$. We enforce that the maximum constraint violation $\gamma(\tau, \mathcal{M}(\sigma))$ is below a given $\epsilon > 0$. Any probabilistically complete single-mode

constrained motion planner [43, 2, 25] PLANCONSTRAINED can be used to plan within extrusion modes. Finally, let $\mathcal{T}(\alpha, \sigma_{\vec{e}}) = \{q \in \mathcal{Q} \mid f_p(q) = p_n, f_o(q) = x_o\}$ denote the set of *unidirectional* transition configurations from the transit mode to extrusion mode $\sigma_{\vec{e}}$, and $\mathcal{T}(\sigma_{\vec{e}}, \alpha) = \{q \in \mathcal{Q} \mid f_p(q) = p_{n'}, f_o(q) = x_o\}$ denote directed transition configurations from extrusion mode $\sigma_{\vec{e}}$ to the transit mode.

C. Extrusion Problems

Definition 2. An *extrusion problem* $\Pi = \langle N, G, E, \mathcal{Q}, q_0 \rangle$ is defined by a set of nodes N, ground nodes G, elements E, configuration space $\mathcal{Q}$, and configuration $q_0 \in \mathcal{Q}$ specifying both the initial and final robot configuration.

Definition 3. For a given error threshold $\epsilon > 0$, a *solution* to an extrusion problem Π is a valid extrusion sequence $\vec{\psi} = [\vec{e}_1, \vec{e}_2, ..., \vec{e}_m]$ (Definition 1), a sequence of extrusion mode coparameters $\vec{\sigma} = [\sigma_{\vec{e}_1}, ..., \sigma_{\vec{e}_m}]$, and an alternating sequence of $m+1$ transit and m extrusion trajectories $\pi = [\tau_{t_1}, \tau_{\vec{e}_1}, ..., \tau_{t_{m+1}}]$ such that:

- $\tau_{t_1}(0) = \tau_{t_{m+1}}(1) = q_0$
- $\forall i \in \{1, ..., m\}$.
  - $\tau_{t_i}(1) = \tau_{\vec{e}_i}(0)$
  - $\forall \lambda \in [0, 1]$. $\tau_{t_i}(\lambda), \tau_{\vec{e}_i}(\lambda) \in Q(\psi_{1:i-1})$
  - $\gamma(\tau_{\vec{e}_i}, \mathcal{M}(\sigma_{\vec{e}_i})) < \epsilon$
- $\tau_{\vec{e}_m}(1) = \tau_{t_{m+1}}(0)$
- $\forall \lambda \in [0, 1]$. $\tau_{t_{m+1}}(\lambda) \in Q(E)$.

V. ALGORITHMIC TOOLS

We present state-space search algorithms for solving extrusion planning problems. States $s = \langle P, q \rangle \in \mathcal{P}(E) \times \mathcal{Q}$ consist of the set of currently printed elements and the current robot configuration where $\mathcal{P}(E)$ denotes the power set of E. The initial state is $s_0 = \langle \emptyset, q_0 \rangle$ and the goal state is $s_* = \langle E, q_0 \rangle$. The PROGRESSION algorithm (Section VII) performs a forward search from $s_0 \to s_*$, and the REGRESSION algorithm (Section VII) performs a backward search from the goal state $s_* \to s_0$. Both PROGRESSION and REGRESSION perform a greedy best-first search [39] guided by a priority function $k(\eta)$ defined over *search nodes* η. On each iteration, the search node η in the *open list* O that minimizes $k(\eta)$ is expanded.

The key trade off when designing these algorithms is the impact on satisfying stiffness and geometric constraints when searching forwards versus backwards. For each constraint in isolation, it is advantageous to search from the *most constrained* state to the *least constrained* state. At a less constrained state, the planner has more options and may prematurely make a decision that limits the legal options later in the search. In contrast, the forward or backward branching factor is generally small at the most constrained state, limiting the availability of poor choices. Additionally, if the constrainedness either provably or empirically decreases over time, the pool of options will grow as the difficulty decreases. Our algorithms leverage this principle, to search in directions that reduce the presence of *dead ends*, because in many extrusion problems, escaping dead ends can require an enormous amount of *backtracking* due to the long planning horizon. We begin by developing common infrastructure for both the PROGRESSION and REGRESSION algorithms.

A. Sampling Extrusions

The key subroutine within each algorithm is SAMPLEEXTRUSION (Algorithm 1), which leverages PLANCONSTRAINED to sample extrusion plans for an element e. First, it samples a start node n_1 based on the currently printed nodes N_P. This governs the extrusion direction $\vec{e} = \langle n_1, n_2 \rangle$. Next, it samples an extrusion mode coparameter $\sigma_{\vec{e}} = x_o$ using SAMPLEORIENTATION. This orientation produces the initial end-effector pose $\langle p_{n_1}, x_o \rangle$ and final end-effector pose $\langle p_{n_2}, x_o \rangle$. Then, we use SAMPLEIK, an inverse kinematics procedure, to sample robot configurations q_1, q_2 that are kinematic solutions for these poses. Finally, we call PLANCONSTRAINED to find a trajectory from $q_1 \to q_2$ that satisfies mode constraints $\sigma_{\vec{e}}$ and does not collide with printed elements P.

Algorithm 1 Extrusion Sampling Algorithm

```
1: procedure SAMPLEEXTRUSION(e, P; i)
2:     n_1 ← sample({n ∈ e | n ∈ N_P})
3:     {n, n'} ← e
4:     n_2 ← n' if n_1 = n else n
5:     x_o ← SAMPLEORIENTATION(n_1, n_2)
6:     q_1 ← SAMPLEIK(p_{n_1}, x_o); q_2 ← SAMPLEIK(p_{n_2}, x_o)
7:     return PLANCONSTRAINED(q_1, q_2, x_o, P; i)
```

B. Deferred Evaluation

Standard state-space searches evaluate all feasible successor states $s' = \langle P \cup \{e\}, q' \rangle$ when expanding a state $s = \langle P, q \rangle$. For extrusion planning, this requires planning both an extrusion trajectory τ_e, where $q' = \tau_e(0)$, and a transit trajectory τ_t from $q \to q'$ for each remaining candidate element $e \in (E \backslash P)$. In the worst case, the number of successor (*i.e.* the branching factor) could be $\mathcal{O}(|E|)$. This is exacerbated due to the fact that SAMPLEEXTRUSION and PLANMOTION are both computationally expensive due to collision-checking. To mitigate this problem, we adopt a *deferred evaluation* [19, 38] strategy by planning extrusion and transit trajectories *after* popping a search node off the open list instead of *before* pushing the node on the open list. To enable this, search nodes in the open list are state and element pairs $\eta = \langle s, e \rangle$ where e serves as "action type" that specifies the next element to be extruded. This strategy dramatically reduces computation time, particularly in a greedy search, because it often avoids checking the feasibility of printing each successor element. Once a feasible successor s' is identified, the yet-to-be evaluated successors are deferred until the greedy search backtracks.

C. Heuristic Tiebreakers

Because search nodes are state and element pairs, the priority function $k(s, e)$ can take the next element e into consideration. We propose priority function $k(\langle P, q \rangle, e) = \langle r(P), h(e) \rangle$ that first orders search nodes by the number of *remaining elements* $r(P) = |E \setminus P|$ and lexicographically breaks ties

using a *heuristic function* $h(e)$ defined on each individual element e. By prioritizing search nodes where few elements remain to be planned, the search greedily explores the state-space in a *depth-first* manner. Because all successor states s' of state s have the same number of remaining elements r, the heuristic tiebreaker decides the order in which successors are considered. This local ordering can have strong global effects on the sequence of partially-extruded structures considered. We consider four implementations of $h(e)$: (1) *Random*, (2) *EuclideanDist* and *GraphDist*, and (3) *StiffPlan*.

1) Random Heuristic: The *Random* tiebreaker is a baseline where ties are broken arbitrarily. It orders elements by assigning each a value sampled uniformly at random $h(e) \sim U(0, 1)$.

2) Distance Heuristics: The *EuclideanDist* and *GraphDist* heuristics prioritize elements that are close to ground, each according to a particular geodesic. The *EuclideanDist* heuristic computes the Euclidean distance from the midpoint of element $e = \{n, n'\}$ to the ground plane. When the ground plane is the xy-plane, this is simply the z-coordinate of the element's midpoint $h_e(e) = (p_n + p_{n'})/2 \cdot [0, 0, 1]^\intercal$. The *GraphDist* heuristic computes the minimum graph distance from any ground node $n \in G$ to the midpoint of element e within the weighted frame geometric graph $\langle N, G\rangle$, where the weight of edge $e = \{n, n'\}$ is the Euclidean distance $||p_n - p_{n'}||$. We precompute these distances upfront once by calling Dijkstra's algorithm starting from the set of ground nodes G. Intuitively, both of these heuristics guide the search through structures where the element load force has a short transfer path to ground because these structures are often stiff. Additionally, these heuristics improve the sample complexity of SAMPLEORIENTATION because they often ensure end-effector orientations opposite to the z-axis remain feasible.

3) Stiffness Heuristic: The *StiffPlan* heuristic solves for a valid extrusion sequence $\vec{\psi}$, *ignoring* the robot, and uses the index j of each element e in the sequence ($\vec{\psi}[j] = e$) as its value $h_s(e) = j$. Intuitively, because $\vec{\psi}$ is known to be stiff, it attempts to adhere to $\vec{\psi}$ as closely as possible subject to the additional robot constraints. We compute a valid extrusion sequence $\vec{\psi}$ using a greedy forward search that is equivalent to PROGRESSION in Algorithm 2 if all robot planning is skipped. We use the *EuclideanDist* heuristic h_e (Section V-C2) as the tiebreaker for this search. See the extended manuscript[1] for the full PLANSTIFFNESS pseudocode.

The *EuclideanDist*, *GraphDist*, and *StiffPlan* heuristics each perform a *forward* computation from ground to produce their values. As we will see in Section VII-B, moving in a forward direction proves to be advantageous for satisfying the stiffness constraint. Finally, these heuristics can be seen as applying "soft" partial-ordering constraints that steer the search but do not limit completeness. This is in contrast to the hard partial-ordering constraints in prior work [47, 20, 22] (Section II).

D. Persistence

The procedures SAMPLEEXTRUSION and PLANMOTION use sampling-based algorithms and thus are are unable to prove infeasibility. As a result, both procedures must be reattempted indefinitely and with an increasing number of samples i. In order to ensure that PROGRESSION and REGRESSION are probabilistically complete, they both are *persistent* [9] searches, meaning that they repeatedly expand each search node in a round-robin fashion. Let $i \geq 0$ denote the number of times a search node has been expanded. We implement persistence by simply using the pair $\langle i, k(s, e)\rangle$ as the key for search nodes in the open list O. This ensures that the search node with the fewest attempts is always expanded first. After a search node is expanded, it is re-added to the search queue O with priority $i + 1$. This search node will not be re-expanded until all other nodes in O have been expanded i times.

[1]The extended version of this manuscript: https://arxiv.org/abs/2002.02360.

VI. PROGRESSION

Algorithm 2 Progression Algorithm

```
1:  procedure PROGRESSION(N, G, E, Q, q_0; h)
2:      O = [⟨0, ⟨|E|, h(e)⟩, ⟨∅, q_0⟩, e, [ ]⟩ for e ∈ E if e ∩ G ≠ ∅]
3:      while True do
4:          i, ⟨r, _⟩, ⟨P, q⟩, e, π ← pop(O)
5:          P' ← P ∪ {e}
6:          if not STIFF(G, P') then
7:              continue                                  ▷ No successors
8:          τ_e ← None
9:          if FORWARDCHECK(E, G, P'; i) then             ▷ Optional
10:             τ_e ← SAMPLEEXTRUSION(e, P; i)            ▷ Extrusion
11:         if τ_e ≠ None then
12:             τ_t ← PLANMOTION(q, τ_e(0), P; i)         ▷ Transit
13:             if τ_t ≠ None then
14:                 π' ← π + [τ_t, τ_e]
15:                 if P' = E then                        ▷ All printed
16:                     τ_t ← PLANMOTION(τ_e(1), q_0, E; i)
17:                     if τ_t ≠ None then
18:                         return π' + [τ_t]             ▷ Solution
19:                 s' ← ⟨P', τ_e(1)⟩
20:                 for e' ∈ (E \ P') do
21:                     push(O, ⟨0, ⟨r − 1, h(e')⟩, s', e', π'⟩)
22:         push(O, ⟨i + 1, ⟨r, h(e)⟩, ⟨P, q⟩, e, π⟩)     ▷ Persistence
```

Algorithm 2 displays the pseudocode for PROGRESSION. Let π be the currently planned trajectories for a search node. After popping a state $\langle P, q\rangle$ and next element e from the open list O, PROGRESSION first checks whether the new structure $P' = P \cup \{e\}$ is stiff, taking advantage of the computational cheapness of STIFF. If not, the search node can be pruned altogether. Otherwise, SAMPLEEXTRUSION samples an extrusion trajectory τ_e for element e. The initial configuration $\tau_e(0)$ then becomes the goal for a transit motion that is found using PLANMOTION. If $P' = E$, then the structure is fully printed, and all that remains is for the robot to return to q_0. Otherwise, all remaining elements $e' \in (E \setminus P')$ are added to O as successor search nodes. Finally, search node $\langle P, q\rangle, e$ is re-added to O with sampling timeout $i + 1$ to be re-expanded in the future (Section V-D). In the extended manuscript[1], we prove PROGRESSION is probabilistically complete.

PROGRESSION is geometrically sensitive to the extrusion sequence ψ. By equation 3, when elements are added to $P = \{e \in \psi\}$, the collision-free configuration space $Q(P)$ weakly

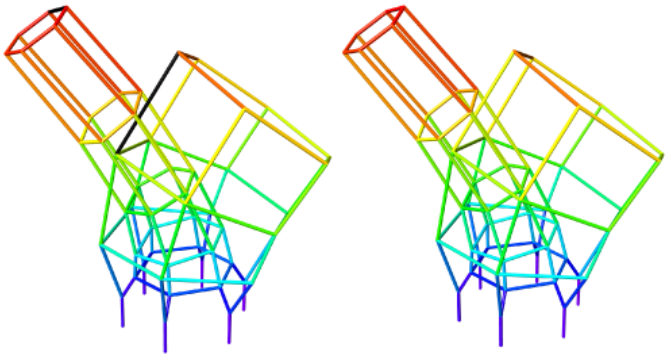

Fig. 3. *Left*: The first state where PROGRESSION-*EuclideanDist* backtracks (black elements are unprinted). *Right*: REGRESSION-*EuclideanDist* finds a solution without backtracking. In our structural figures, elements are colored by their index in a planned extrusion sequence. Purple elements are printed first, red elements are printed last, and black elements have yet to be printed.

decreases, causing SAMPLEEXTRUSION and PLANMOTION to become more constrained. In the worst case, P may prevent some of the unprinted elements $E \setminus P$ from admitting any safe extrusions. For example, Figure 3 demonstrates that PROGRESSION becomes trapped in a dead end near the end of the horizon because it printed the left tail of the Klein bottle (Figure 1) before the black diagonal element.

A. Forward Checking for Dead-End Detection

In order to help PROGRESSION avoid making poor geometric decisions, we developed a *forward-checking* (look ahead) Algorithm [16, 6] that is able to detect dead ends earlier in the search. Intuitively, the robot must extrude every element in the structure eventually. If there is ever an element that cannot be extruded given the partially-extruded structure P, then this state is a dead end. Thus, FORWARDCHECK eagerly evaluates the viability of many successors. However, this acts oppositely to deferred evaluation (Section V-B), and thus achieves better dead-end detection at the expense of worse computational overhead. As a compromise, we plan *extrusion trajectories* for only the elements e that can *currently* can be printed given P, (*i.e.* $e \cap N_P \neq \emptyset$). Intuitively, these elements are close in proximity to the printed structure and thus are most likely to be affected by a proposed geometric decision.

Algorithm 3 displays the pseudocode for FORWARDCHECK. It maintain a global *cache* of extrusion trajectories in order to reuse previously computed trajectories if possible. Because FORWARDCHECK invokes SAMPLEEXTRUSION, it cannot prove that a search node is a dead end. Thus, FORWARDCHECK also uses the increasing sampling timeout i to search for longer extrusion trajectories. Figure 4 demonstrates an instance where FORWARDCHECK detects, and thus avoids, a dead end early in the search. The element with the pink sphere is the candidate element e to be printed. However, printing e prevents the diagonal black element from being printable. As a result, the search defers expanding e at this time.

FORWARDCHECK performs a one-step look ahead to detect dead ends. However, it might the case that while each element

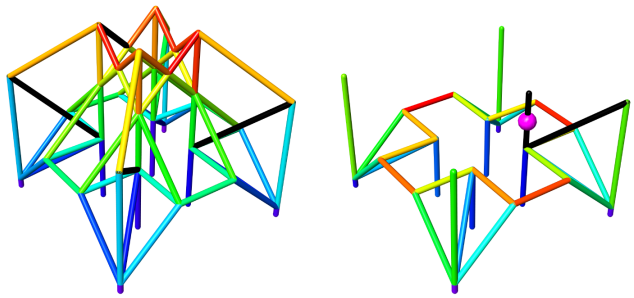

Fig. 4. *Left*: the first state where PROGRESSION-*GraphDist* backtracks (black elements are unprinted). *Right*: FORWARDCHECK detects that printing the element indicated by the pink sphere prevents the diagonal black element from being safely extrudable.

Algorithm 3 Forward Checking Algorithm

```
1:  procedure ForwardCheck(E, G, P; i)
2:      cache ← {e : [ ] for e ∈ E}                    ▷ Global cache
3:      for e ∈ (E \ P) do
4:          if e ∩ N_P = ∅ then                        ▷ Printable
5:              continue
6:          if any(Safe(τ_e, P) for τ_e ∈ cache[e]) then
7:              continue                               ▷ Reuse existing
8:          τ_e ← SampleExtrusion(e, P; i)             ▷ Extrusion
9:          if τ_e = None then
10:             return False
11:         cache[e] ← cache[e] + [τ_e]
12:     return True
```

can be printed individually, a *pair* of elements together cannot be printed. If so, FORWARDCHECK will not be able to detect the dead end until much later in the search, such shown in Figure 5. Here, extruding any black element prevents at least one other nearby element from being safely printable. An *arc-consistency* look ahead that considers pairs [40] could detect these cases at the expense of even greater expansion overhead.

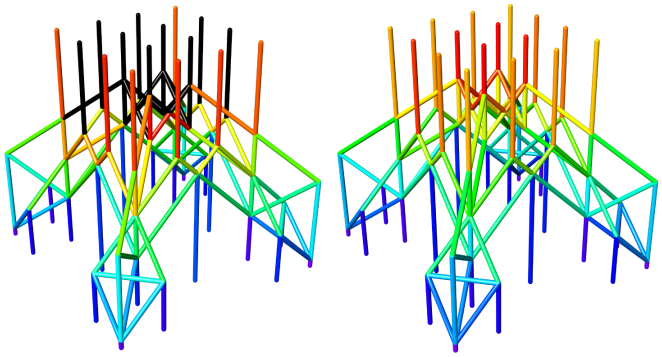

Fig. 5. *Left*: the first state where FORWARDCHECK-*GraphDist* backtracks (black elements are unprinted). *Right*: REGRESSION-*EuclideanDist* finds a solution without backtracking.

VII. REGRESSION

REGRESSION performs a backward search from the goal state to the initial state [36, 46, 34, 13]. In many planning domains, the goal conditions are under-specified, and as a

result, there are many goal states. Because of this, the initial branching factor can be quite large. Furthermore, some goal states might not be reachable from s_0, creating more opportunities for dead-end branches [3]. Because extrusion planning has a single goal state s_*, these problems are avoided. Algorithm 4 displays the pseudocode for REGRESSION. The key differences from PROGRESSION in Algorithm 2 are that we negate $-h(e)$ in order to expand elements in the reverse order, the final extrusion configuration $\tau_e(1)$ is the start of each transit motion planning problem, and trajectories $[\tau_e, \tau_t]$ are prepended to plan π. In the extended manuscript[1], we prove REGRESSION is probabilistically complete.

Algorithm 4 Regression Algorithm

procedure REGRESSION($N, G, E, \mathcal{Q}, q_0; h$)
  $O = [\langle 0, \langle |E|, -h(e)\rangle, \langle E, q_0\rangle, e, [\,]\rangle$ **for** $e \in E]$
  while True do
    $i, \langle r, \_\rangle, \langle P, q\rangle, e, \pi \leftarrow$ **pop**(O)
    $P' \leftarrow P \setminus \{e\}$
    if not STIFF(G, P') **then**
      continue ▷ No successors
    $\tau_e \leftarrow$ SAMPLEEXTRUSION$(e, P'; i)$ ▷ Extrusion
    if $\tau_e \neq$ **None then**
      $\tau_t \leftarrow$ PLANMOTION$(\tau_e(1), q, P; i)$ ▷ Transit
      if $\tau_t \neq$ **None then**
        $\pi' \leftarrow [\tau_e, \tau_t] + \pi$
        if $P' = \emptyset$ **then** ▷ All printed
          $\tau_t \leftarrow$ PLANMOTION$(q_0, \tau_e(0); \emptyset, i)$
          if $\tau_t \neq$ **None then**
            return $[\tau_t] + \pi'$ ▷ Solution
        $s' \leftarrow \langle P', \tau_e(0)\rangle$
        for $e' \in P'$ **do**
          push$(O, \langle 0, \langle r-1, -h(e')\rangle, s', e', \pi'\rangle)$
    push$(O, \langle i+1, \langle r, -h(e)\rangle, \langle P, q\rangle, e, \pi\rangle)$ ▷ Persistence

A. Geometric Constraints

REGRESSION can be seen as *deconstructing* the structure by sequentially removing elements. From equation 3, removing an element weakly increases the collision-free configuration space $Q(P)$. Thus, the robot is the most geometrically constrained at the beginning of the search, limiting which elements can be initially extruded. As a result, REGRESSION's options with respect to geometry increase as the search advances, preventing it from being trapped in a geometric dead end. To motivate using backward search to efficiently satisfy geometric constraints, we analyze a simplified *geometry-only* version of the extrusion problem that both omits stiffness and transit constraints as well as assumes a given set of possible extrusion trajectories T. Given these simplifications, extrusion planning simply requires identifying a totally-ordered subset of T that extrudes each element exactly once. We consider a modified version of REGRESSION in Algorithm 4 for extrusion-only problems. Trivially, for all inputs, let STIFF(G, P) = **True** and PLANMOTION$(q, q', P; i) = [q, q']$. Additionally, SAMPLEEXTRUSION$(e, P; i)$ = **sample**$(\{\tau_e \in T \mid$ SAFE$(\tau_e, P)\})$ arbitrarily selects a safe trajectory $\tau_e \in T$ for element e if one exists. Otherwise, **sample** returns **None**. Under these conditions, REGRESSION will solve feasible problem instances in polynomial time (see the extended manuscript[1]).

B. Stiffness Constraints

Although REGRESSION makes geometric planning easier, it increases the difficulty of satisfying the stiffness constraint. At the beginning of the backward search, there are many elements that can be removed without violating the stiffness constraint. However, later in the backward search (closer to the structure's supports), there are fewer opportunities for supporting the structure, making the search more likely to arrive at a dead end caused by stiffness. Figure 6 image 1) shows the remaining-to-be-printed structure at the first dead end encountered by REGRESSION-*Stiffness*. As can be seen, arbitrarily removing elements sparcifies the structure and reduces its structural integrity. To combat this, we use the heuristic tiebreakers in Section V-C to bias the search to remain stiff.

To understand the impact of these tiebreakers, we experimented on the extrusion problems in Section VIII, comparing the success rate of the PROGRESSION and REGRESSION algorithms when *only* the stiffness constraint is active (*i.e.* ignoring the robot). For PROGRESSION, this is equivalent to PLANSTIFFNESS in Section V-C3. We performed 6 trials per algorithm, heuristic, and problem. Each trial had a 5 minute timeout. Figure 7 displays the success rate of each algorithm. PROGRESSION was able to find an extrusion sequence for all problems, regardless of the heuristic. REGRESSION failed around 40% of the time when randomly breaking ties. However, REGRESSION was able to solve all problems when using the *StiffPlan* heuristic; although, this is not surprising given that *StiffPlan* explicitly uses a stiff plan. The *EuclideanDist* and *GraphDist* heuristics perform quite well but still have failure cases, such as in Figure 6. There, both heuristics prioritize removing the top of the structure, which is designed to provide tensile forces to hold the cantilevered elements [33], causing the red vertices to deform significantly.

VIII. RESULTS

We experimented on 41 extrusion problems with up to 909 elements (the duck problem in Figure 1). See the extended manuscript[1], for a picture of each problem. We experimented using all combinations of our 3 algorithms (PROGRESSION, FORWARDCHECK, and REGRESSION) and 4 heuristics (*Random*, *EuclideanDist*, *GraphDist*, and *StiffPlan*). We performed 4 trials per algorithm, heuristic, and problem, each with a 1 hour timeout. We used PyBullet [4, 5] for collision checking, forward kinematics, and rendering. Because each element can only be in one pose, we preprocess the structure by computing a single, static axis-aligned bounding box (AABB) bounding volume hierarchy (BVH) [8, 27] for use during broadphase collision detection with each robot link. We implemented PLANMOTION using RRT-Connect [29], SAMPLEIK using IKFast, an analytical inverse kinematics solver [7], and PLANCONSTRAINED using Randomized Gradient Descent (RGD)[48, 43]. See https://github.com/caelan/pb-construction for implementations of our algorithms.

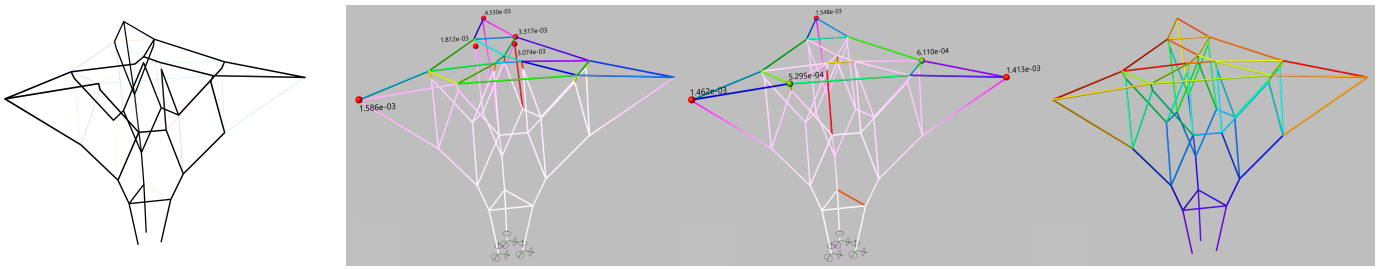

Fig. 6. *From left to right*: 1) the unassigned substructure at the first state where REGRESSION-*Random* backtracks. 2) the first state where REGRESSION-*EuclideanDist* backtracks. The element *deflection* is colored from white to pink. The five most deformed nodes are red and their translational displacements are annotated in meters 3) the first state where REGRESSION-*GraphDist* backtracks 4) REGRESSION-*StiffPlan* finds a solution without backtracking.

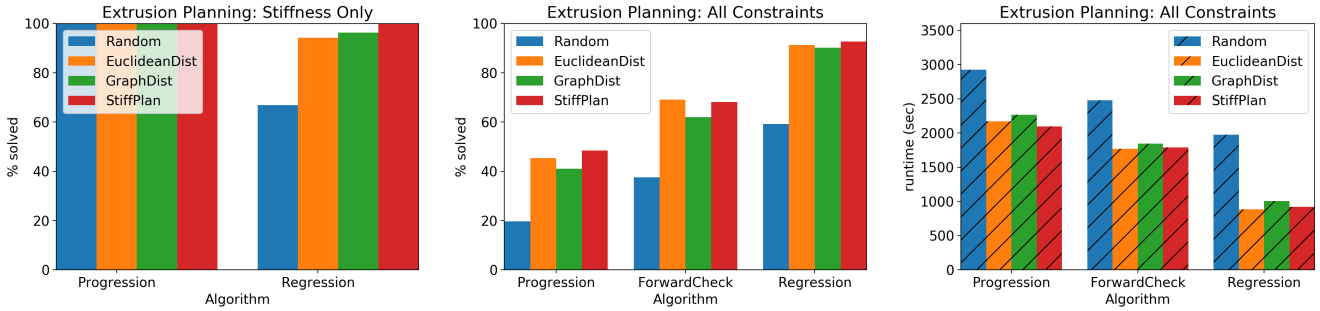

Fig. 7. *Left*: the success rate of each algorithm (except FORWARDCHECK) and heuristic pair subject to *only the stiffness constraint*. *Center*: the success rate of each algorithm and heuristic pair. *Right*: the average runtime in seconds of each algorithm and heuristic pair with a timeout of 1 hour (3600 seconds).

Figure 7 displays the success rate (*Center*) and the average runtime (*Right*) for each algorithm. We assign a runtime of 1 hour for trials that failed to find a solution. The *EuclideanDist*, *GraphDist*, and *StiffPlan* heuristics outperform *Random*, regardless of the algorithm. The improved performance for both PROGRESSION and REGRESSION indicates that the heuristics provide both stiffness and geometric guidance. FORWARDCHECK is able to solve more problems than PROGRESSION, indicating that it is able to avoid some dead ends. However, ultimately REGRESSION performed the best in terms of both success rate and runtime. The best performing heuristic was *StiffPlan* followed closely by the *EuclideanDist*. Our best-performing algorithms are able to solve around 92% of the problems and have an average runtime of about 15 minutes. Figure 8 (*Right*) displays the runtime of each trial per problem size when each algorithm uses the *EuclideanDist* heuristic. Although FORWARDCHECK is able to solve more problems than PROGRESSION, it comes at the expense of longer runtimes.

We experimented on two extrusion problems considered by Choreo [22]. Choreo solves the "3D Voronoi" and "Topopt beam (small)" problems in 4025 and 3599 seconds whereas REGRESSION-*EuclideanDist* solves the problems in 742 and 2032 seconds. Our planner outperforms Choreo despite the fact that Choreo had access to additional, human-specified information (Section II). We validated our approach on three real-world extrusion problems. See https://youtu.be/RsBzc7bEdQg for a video of our robot extruding each structure. The largest

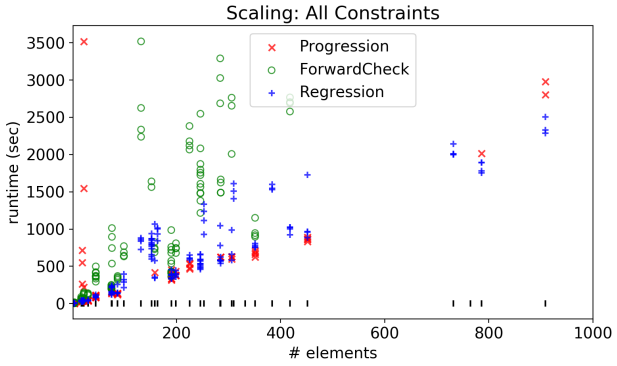

Fig. 8. The runtime of each algorithm when using the *EuclideanDist* heuristic. The x-axis ticks denote the distribution of problem sizes.

of the three is the Klein bottle (Figure 1), which took about 10 minutes to plan for and 6 hours to print.

IX. CONCLUSION

We investigated 3D extrusion planning using a robot manipulator. Here, structural constraints are often at odds with geometric constraints. Our algorithmic insight was to use backward search to plan geometrically feasible trajectories and to use forward reasoning as a heuristic that guides the search through structurally-sound states. Future work involves extending our approach to general-purpose construction tasks.

References

[1] R Alami, J.-P. Laumond, and T Siméon. Two manipulation planning algorithms. In *Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1994. URL http://dl.acm.org/citation.cfm?id=215085.

[2] Dmitry Berenson, Siddhartha Srinivasa, and James Kuffner. Task space regions: A framework for pose-constrained manipulation planning. *The International Journal of Robotics Research*, 30(12):1435–1460, 2011.

[3] Blai Bonet and Héctor Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1):5–33, 2001.

[4] Erwin Coumans. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*, page 7. ACM, 2015.

[5] Erwin Coumans and Yunfei Bai. PyBullet, a Python module for physics simulation for games, robotics and machine learning. \url{http://pybullet.org}, 2016.

[6] Rina Dechter. *Constraint processing*. Morgan Kaufmann, 2003.

[7] Rosen Diankov. *Automated construction of robotic manipulation programs*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2010.

[8] Christer Ericson. *Real-time collision detection*. CRC Press, 2004.

[9] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Backward-Forward Search for Manipulation Planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2015-Decem, 2015. ISBN 9781479999941. doi: 10.1109/IROS.2015.7354287. URL http://lis.csail.mit.edu/pubs/garrett-iros15.pdf.

[10] Caelan Reed C.R. Garrett, Tomás Lozano-Pérez, and L.P. Leslie Pack Kaelbling. Sampling-based methods for factored task and motion planning. *The International Journal of Robotics Research*, 37(13-14), 2018. ISSN 17413176. doi: 10.1177/0278364918802962. URL https://arxiv.org/abs/1801.00680.

[11] Matthew K Gelber, Greg Hurst, and Rohit Bhargava. Freeform Assembly Planning. *arXiv:1801.00527*, 1 2018.

[12] J M Gere and S P Timoshenko. Mechanics of materials, 1997. *PWS-KENT Publishing Company, ISBN 0*, 534 (92174):4, 1997.

[13] Malik Ghallab, Dana S Nau, and Paolo Traverso. *Automated Planning: Theory and Practice*. Elsevier, 2004.

[14] Norman Hack and Willi Viktor Lauer. Mesh-Mould: Robotically Fabricated Spatial Meshes as Reinforced Concrete Formwork. *Architectural Design*, 84(3):44–53, 2014.

[15] Shuai D Han, Nicholas M Stiffler, Athanasios Krontiris, Kostas E Bekris, and Jingjin Yu. Complexity results and fast methods for optimal tabletop rearrangement with overhand grasps. *The International Journal of Robotics Research*, 37(13-14):1775–1795, 2018.

[16] Robert M Haralick and Gordon L Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial intelligence*, 14(3):263–313, 1980.

[17] Kris Hauser and Jean-Claude Latombe. Multi-modal Motion Planning in Non-expansive Spaces. *International Journal of Robotics Research (IJRR)*, 29:897–915, 2010.

[18] Kris Hauser and Victor Ng-Thow-Hing. Randomized multi-modal motion planning for a humanoid robot manipulation task. *International Journal of Robotics Research (IJRR)*, 30(6):676–698, 2011. URL http://journals.sagepub.com/doi/abs/10.1177/0278364910386985.

[19] Malte Helmert. The Fast Downward Planning System. *Journal of Artificial Intelligence Research (JAIR)*, 26:191–246, 2006. URL http://www.jair.org/papers/paper1705.html.

[20] Yijiang Huang, Juyong Zhang, Xin Hu, Guoxian Song, Zhongyuan Liu, Lei Yu, and Ligang Liu. Framefab: Robotic fabrication of frame shapes. *ACM Transactions on Graphics (TOG)*, 35(6):224, 2016.

[21] Yijiang Huang, Josephine Carstensen, and Caitlin Mueller. 3D truss topology optimization for automated robotic spatial extrusion. In *Proceedings of the International Association for Shell and Spatial Structures (IASS) Symposium 2018*. 2018.

[22] Yijiang Huang, Caelan R Garrett, and Caitlin T Mueller. Automated sequence and motion planning for robotic spatial extrusion of 3D trusses. *Construction Robotics*, 2(1):15–39, 12 2018. ISSN 2509-8780. doi: 10.1007/s41693-018-0012-z. URL https://doi.org/10.1007/s41693-018-0012-z.

[23] Sertac Karaman and Emilio Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. *International Journal of Robotics Research (IJRR)*, 30(7):846–894, 2011.

[24] Lydia E Kavraki, Mihail N Kolountzakis, and J-C Latombe. Analysis of probabilistic roadmaps for path planning. *Robotics and Automation, IEEE Transactions on*, 14(1):166–171, 1998.

[25] Zachary Kingston, Mark Moll, and Lydia E. Kavraki. Exploring implicit spaces for constrained sampling-based planning. *International Journal of Robotics Research*, 38 (10-11):1151–1178, 2019. ISSN 17413176. doi: 10.1177/0278364919868530.

[26] M Kleinbort, K Solovey, Z Littlefield, K E Bekris, and D Halperin. Probabilistic Completeness of RRT for Geometric and Kinodynamic Planning With Forward Propagation. *IEEE Robotics and Automation Letters*, 4 (2):x–xvi, 4 2019. ISSN 2377-3774. doi: 10.1109/LRA.2018.2888947.

[27] Daniel Kopta, Thiago Ize, Josef Spjut, Erik Brunvand, Al Davis, and Andrew Kensler. Fast, effective BVH updates for animated scenes. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 197–204. ACM, 2012.

[28] A Krontiris and K E Bekris. Dealing with Difficult Instances of Object Rearrangement. In *Robotics: Science and Systems (RSS)*, Rome, Italy, 9 2015. URL http://www.cs.rutgers.edu/~kb572/pubs/Krontiris_Bekris_rearrangement_RSS2015.pdf.

[29] James J Kuffner Jr. and Steven M LaValle. {RRT-Connect}: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[30] F. Lagriffoul, N.T. Dantam, C. Garrett, A. Akbari, S. Srivastava, and L.E. Kavraki. Platform-Independent Benchmarks for Task and Motion Planning. *IEEE Robotics and Automation Letters*, 3(4), 2018. ISSN 23773766. doi: 10.1109/LRA.2018.2856701.

[31] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[32] Steven M LaValle. *Planning Algorithms*. Cambridge University Press, 2006. URL msl.cs.uiuc.edu/planning.

[33] Juney Lee. *Computational Design Framework for 3D Graphic Statics*. PhD thesis, ETH Zurich, Department of Architecture, Zurich, 2018.

[34] Drew McDermott. Regression planning. *International Journal of Intelligent Systems*, 6(4):357–416, 1991.

[35] W McGuire, R H Gallagher, and R D Ziemian. *Matrix Structural Analysis*. Wiley, 1999. ISBN 978-0-471-12918-9.

[36] Nils J Nilsson. *Principles of artificial intelligence*. Morgan Kaufmann, 2014.

[37] Daniel Piker and Richard Maddock. Continuous Robotic Spatial 3D Printing of Topologically Irregular Space Frames. In Christoph Gengnagel, Olivier Baverel, Jane Burry, Mette Ramsgaard Thomsen, and Stefan Weinzierl, editors, *Impact: Design With All Senses*, pages 502–516, Cham, 2020. Springer International Publishing. ISBN 978-3-030-29829-6. doi: 10.1007/978-3-030-29829-6{_}39.

[38] Silvia Richter and Malte Helmert. Preferred Operators and Deferred Evaluation in Satisficing Planning. In *ICAPS*, 2009.

[39] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[40] Daniel Sabin and Eugene C Freuder. Contradicting conventional wisdom in constraint satisfaction. In *International Workshop on Principles and Practice of Constraint Programming*, pages 10–20. Springer, 1994.

[41] Thierry Siméon, Jean-Paul Laumond, Juan Cortés, and Anis Sahbani. Manipulation planning with probabilistic roadmaps. *International Journal of Robotics Research (IJRR)*, 23(7-8):729–746, 2004.

[42] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined Task and Motion Planning Through an Extensible Planner-Independent Interface Layer. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[43] Mike Stilman. Global manipulation planning in robot joint space with task constraints. *IEEE Transactions on Robotics*, 26(3):576–584, 2010.

[44] Kam-Ming Tam, Daniel J M Marshall, Mitchell Gu, Jasmine Kim, Yijiang Huang, Justin A Lavallee, and Caitlin T Mueller. Fabrication-aware structural optimisation of lattice additive-manufactured with robot-arm. *International Journal of Rapid Manufacturing*, 7(2-3), 2018.

[45] Marc Toussaint and Manuel Lopes. Multi-bound tree search for logic-geometric programming in cooperative manipulation domains. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4044–4051, 2017. ISBN 9781509046331. doi: 10.1109/ICRA.2017.7989464.

[46] Daniel S Weld. An introduction to least commitment planning. *AI magazine*, 15(4):27, 1994.

[47] Rundong Wu, Huaishu Peng, François Guimbretière, and Steve Marschner. Printing arbitrary meshes with a 5DOF wireframe printer. *ACM Transactions on Graphics (TOG)*, 35(4):101, 2016.

[48] Zhenwang Yao and Kamal Gupta. Path planning with general end-effector constraints. *Robotics and Autonomous Systems*, 55(4):316–327, 2007.

[49] Lei Yu, Yijiang Huang, Zhongyuan Liu, Sai Xiao, Ligang Liu, Guoxian Song, and Yanxin Wang. Highly Informed Robotic 3D Printed Polygon Mesh: A Novel Strategy of 3D Spatial Printing. In *Proceedings of the 36st Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*, pages 298–307, 2016.