A Temporal Coherent Topology Optimization Approach for Assembly Planning of Bespoke Frame Structures

ZIQI WANG, ETH Zurich, Switzerland
FLORIAN KENNEL-MAUSHART, ETH Zurich, Switzerland
YIJIANG HUANG, ETH Zurich, Switzerland
BERNHARD THOMASZEWSKI, ETH Zurich, Switzerland
STELIAN COROS, ETH Zurich, Switzerland

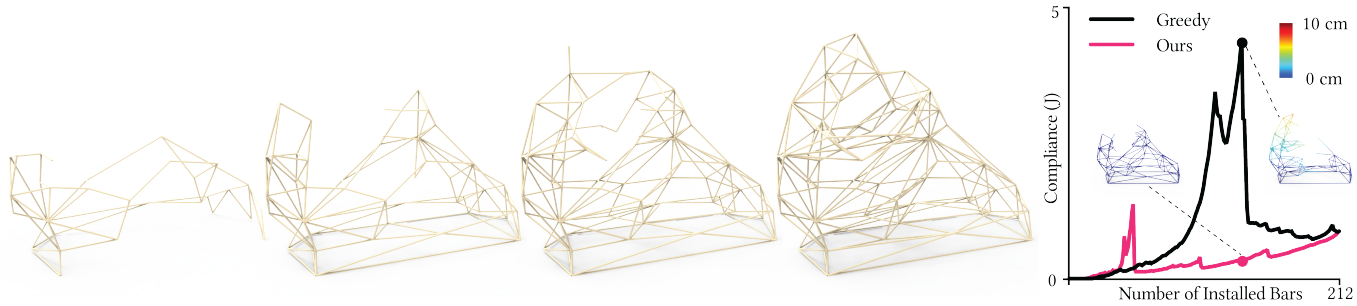

Fig. 1. Our method can automatically plan the assembly sequence for building a sculpture measuring $10m \times 3m \times 7m$, composed of 212 bars. Using a new topology optimization formulation, our method generates three temporal coherent sub-assemblies, which we refer to as "landmarks", and runs a greedy search, which selects the bar at each assembly step that minimizes the structural deformation of the next intermediate substructure, to compute the assembly sequences between these landmarks. The plot illustrates the structural compliance of the intermediate assemblies over the number of installed bars for our method (magenta) and the baseline sequence computed by a greedy search algorithm (black) without using our landmarks. Our approach can plan in the long-term and is able to take more costly steps at the beginning in return for a reduction in assembly cost later. In this example, our sequence reduces the structural compliance by 65% compared to the baseline.

We present a computational framework for planning the assembly sequence of bespoke frame structures. Frame structures are one of the most commonly used structural systems in modern architecture, providing resistance to gravitational and external loads. Building frame structures requires traversing through several partially built states. If the assembly sequence is planned poorly, these partial assemblies can exhibit substantial deformation due to self-weight, slowing down or jeopardizing the assembly process. Finding a good assembly sequence that minimizes intermediate deformations is an interesting yet challenging combinatorial problem that is usually solved by heuristic search algorithms. In this paper, we propose a new optimization-based approach that models sequence planning using a series of topology optimization problems. Our key insight is that enforcing temporal coherent constraints in the topology optimization can lead to sub-structures with small deformations while staying consistent with each other to form an assembly sequence. We benchmark our algorithm on a large data set and show improvements in both performance and computational time over greedy search algorithms. In addition, we demonstrate that our algorithm can be extended to handle assembly with static or dynamic supports. We further validate our approach by generating a series of results in multiple scales, including a real-world prototype with a mixed reality assistant using our computed sequence and a simulated example demonstrating a multi-robot assembly application.

CCS Concepts: • **Theory of computation → Discrete optimization**; • **Computing methodologies → Shape modeling**.

Additional Key Words and Phrases: frame structure, assembly sequence planning, topology optimization

ACM Reference Format:
Ziqi Wang, Florian Kennel-Maushart, Yijiang Huang, Bernhard Thomaszewski, and Stelian Coros. 2023. A Temporal Coherent Topology Optimization Approach for Assembly Planning of Bespoke Frame Structures. *ACM Trans. Graph.* 41, 4, Article xxx (July 2023), 13 pages. https://doi.org/10.1145/8888888.7777777

Authors' addresses: Ziqi Wang, ETH Zurich, Switzerland, ziqi.wang@inf.ethz.ch; Florian Kennel-Maushart, ETH Zurich, Switzerland, florian.maushart@inf.ethz.ch; Yijiang Huang, ETH Zurich, Switzerland, yijiang.huang@inf.ethz.ch; Bernhard Thomaszewski, ETH Zurich, Switzerland, bthomasz@inf.ethz.ch; Stelian Coros, ETH Zurich, Switzerland, stelian.coros@inf.ethz.ch.

0730-0301/2023/07-ARTxxx $15.00
https://doi.org/10.1145/8888888.7777777

1 INTRODUCTION

Assemblies are important to human society as many complex machines and large-scale buildings are assembled from small and simple elements. Recent developments in computational design offer many tools to accelerate the assembly design process and to improve the performance of the final designs [Wang et al. 2019; Whiting

et al. 2012]. However, as size and complexity increase, the cost of assembling a structure becomes more and more significant.

In this work, we focus on assembling bespoke frame structures, which have many applications in civil and mechanical engineering [Apolinarska 2018; Parascho 2019]. Constructing a frame structure involves installing prefabricated bars following a given assembly sequence. Any such sequence inevitably includes incomplete assembly states, and these intermediate states can exhibit large structural deformations due to self-weight. The order in which bars are assembled strongly influences the structural deformation of its intermediate assembly states. Among all feasible assembly sequences, there are generally many poor ones with large structural deformation and a few good ones with smaller deformation. Therefore, an exciting yet computationally challenging problem is to find the optimal assembly sequence that minimizes structural deformations of intermediate states during assembly.

Finding the optimal sequence for constructing a given frame structure is a difficult combinatorial problem. Exhaustively enumerating all feasible sequences is impractical for common computing devices because of the enormous search space. Existing approaches [Huang et al. 2021; McEvoy et al. 2014] use greedy search algorithms to solve sequencing problems. However, as illustrated in Fig 1, the assembly plan generated by a structurally-informed greedy search algorithm can still exhibit unacceptably large deformations.

To address this problem, we propose a new optimization-based approach for solving sequence planning problems in constructing bespoke frame structures. We observe that any assembly sequence can be represented through its intermediate assembly states. Thus, solving sequence planning problems is equivalent to finding a set of temporal coherent intermediate sub-assemblies with minimum structural deformation. For illustration, consider an assembly problem in which half of the bars can be installed simultaneously. Here, finding the optimal sequence is equivalent to computing the stiffest sub-assembly with half of the bars, which is a classic problem in topology optimization. To generalize this strategy to assembling $h \geq 1$ bars at a time, our key insight is to use a new *temporal coherent topology optimization* formulation that jointly solves for several intermediate states to form an optimal assembly sequence. We cast this new formulation as a mixed integer programming problem and propose different strategies for its solution. Specifically, we make the following contributions:

- We propose a new continuous relaxation of discrete sequence planning problems inspired by topology optimization. We solve our new formulation using mixed integer optimization.
- We provide a unified framework for assembling frame structures that install one bar at a time or multiple bars at a time. Our method also supports assembly sequences with non-uniform step sizes.
- Our framework can be extended to handle assembly-with-support (static or dynamic) procedures.
- The assembly sequences generated by our method are agnostic to the assembly process and can be adapted to various setups such as mixed-reality assisted manual assembly or robotic assembly.

We validate our method in a large data set containing 100 frame structures. Compared to the greedy search approach, our method can reduce up to 70% of the structural deformations in the single-bar-at-a-time assembly process and obtain more than 10 times speed improvement in the 6-bars-at-a-time assembly process. We validate our sequence by physically assembling one complex frame structure with the assistance of mixed reality. To illustrate future applications, we show a simulated, multi-robot assembly case study using our computed sequence. Our code and benchmark are available at https://github.com/KIKI007/sequencer_benchmark.

2 RELATED WORK

Frame Structure Construction is a topic that has been widely researched in academia and industry. We only review works that assemble bespoke frame structures through computational approaches. Early works [Doggett 2002] aim to construct frame structures in extraterrestrial environments, where robots replace humans in assembling bars according to a pre-computed sequence. Since then, research projects have treated the construction of many kinds of frame structures, such as furniture [Jacobson 2019], reciprocal frames [Song et al. 2013], roofs [Apolinarska et al. 2016], formwork [Kumar et al. 2017], timber houses [Eversmann et al. 2017; Thoma et al. 2018], and art installations [Brütting et al. 2021; Gandia et al. 2018; Yoshida et al. 2015].

Planning the assembly sequence for constructing a bespoke frame structure can be considered as part of the design process. Architects usually design their frame structures to have natural assembly sequences. For instance, frame structures are designed to be assembled in a layer-by-layer manner [Apolinarska et al. 2016; Leder et al. 2019; Naboni et al. 2021]. Modular design [Eversmann et al. 2017; Jenett et al. 2019] is another widely applied method where modules are fabricated remotely and assembled onsite to form a large frame structure. However, such assembly methods can limit the users' design freedom. Our method is not limited to certain design typologies and can solve sequence planning problems for any input frame structure.

Some recent works in frame structure construction aim to solve the sequence planning problem with general inputs. McEvoy et al. [2014] first compute the assembly sequence that minimizes the structural deformation of the intermediate substructures during construction using a greedy search with backtracking. Huang et al. [2021] employ a robot to assemble frame structures. They propose a forward and backward search framework to minimize the structural deformation of incomplete substructures. However, both works only support assembling one bar at a time. Bruun et al. [2022] compute the assembly sequences for constructing frame structures using multiple robots. They assume all bars are connected by pin joints and can rotate around the joints. Their method ensures that their incomplete substructures are infinitesimally rigid, which prevents the truss structures from having extra degrees of freedom. However, their algorithm is mainly meant for assembling frame structures composed of tetrahedron cells, limiting their input shapes.

Assembly Sequence Planning aims at finding a feasible assembly plan that can be used to assemble a given design without causing deadlocks [Jiménez 2013; Tian et al. 2022; Wang et al. 2021]. Recent works also employ robots for assembly [Hartmann et al. 2020, 2023; Huang et al. 2021], and therefore need to solve both robotic path planning and sequence planning simultaneously. If many feasible sequences exist, we would like to compare the quality of different sequences. Some widely-used measurements include fabrication quality [Wu et al. 2016], uncertainty [Behdad et al. 2014], structural vibration [Wang et al. 2022], structural stability [Deuss et al. 2014] and structural stiffness [Hayashi et al. 2022; Huang et al. 2016]. We focus on the papers concerning structural stability and stiffness. Both objectives measure the structural performance of the intermediate assembly states defined by an assembly plan. The only difference is that structural stability is used for rigid assembly, treated as a collection of rigid blocks (e.g., masonry structure), and structural stiffness is used for deformable assembly, treated as a collection of elastic bodies (e.g., frame structure).

Deuss et al. [2014] present an algorithm for assembling self-supporting shell structures with rigid blocks. They utilize external chains to stabilize the incomplete sub-assemblies during construction temporarily. They propose a divide-and-conquer search method to reduce the number of chains needed for construction, which builds an intermediate sub-assembly first and then adds the remaining blocks to complete the construction. Since constructing frame structures is generally different from assembling masonry shells, directly applying their method to solve our sequencing problem is not trivial.

Huang et al. [2016] propose a sequence planning algorithm for spatially extruding plastic bars using a robot arm equipped with a 3D-printing hotend. Their goal is to find a printing sequence that limits the structural deformation of printed bars while ensuring the hotend has a collision-free path for extrusion. They propose a divide-and-conquer algorithm based on graph cuts to split their final structure into several structurally stiff layers. This layering strategy divides their sequence planning problem into several independent sub-problems, which can be efficiently solved using a local search. However, [Huang et al. 2016] only prints one bar at a time. We also show in the result section that our method can solve their layer decomposition problem with a better performance in terms of structural stiffness.

Topology Optimization is concerned with maximizing the structural stiffness of a design under given external loads within a material budget [Bendsoe and Sigmund 2003]. Among many types of methods in topology optimization, the methods for designing frame/truss structures, i.e. truss topology optimization (TTO), are the most relevant to our work. The design variables of TTO are the cross-section areas of the bars. Previous works can be classified into continuous TTO and discrete TTO depending on the type of design variables. Since our assembly planning is a discrete process, we focus on methods for solving discrete TTO. In the survey paper [Stolpe 2016], global optimization methods (e.g., mixed integer optimization) and stochastic methods (e.g., genetic algorithm) are listed as the two typical ways of solving discrete TTO problems. Typical discrete TTO methods include the primal method [Kocvara 2010], which solves a mixed-integer semi-definite programming, and the dual method [Achtziger and Stolpe 2009], which solves a mixed-integer quadratically-constrained quadratic program.

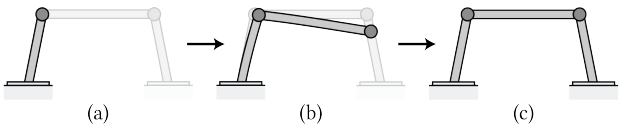

Fig. 2. The assembly procedure for a 3-bar frame structure. The left and right ends of the structure are anchored to the ground. Transparent bars in figures (a) and (b) depict the complete structure for reference purposes. The intermediate structure in (b) has the highest assembly cost during assembly.

Jiang et al. [2017] propose a design framework for generating structurally sound truss structures for architectural applications. Arora et al. [2019] present an algorithm to design volumetric Michell trusses, allowing users to edit their designs parametrically. Neveu [2022] develops a method for simplifying a curve network to minimize deformation under worst-case loads. Most of these works focus on reducing the deformation of completed structures, while our work focuses on reducing the deformation during the assembly process. Previous works [Allaire et al. 2017; Langelaar 2017] suggest additional fabrication constraints, like no overhangs, to enable fabricating topology optimization shapes using standard additive manufacturing methods. Wang et al. [2020] present a time-based topology optimization method for improving the structural stiffness of intermediate structures during a 3D printing process. Lan et al. [2022] proposes a novel computational approach that further improves the computational efficiency of optimizing the structural stiffness during a 3D printing process. However, these methods are mostly designed for 3D printing of continuum shapes. It is not straightforward to adapt their method to solve our sequence planning problem in assembling frame structures.

3 PROBLEM FORMULATION

Let's consider assembling a frame structure that has n elastic bars $\{b_i\}$; see Figure 2. We consider the sequence planning problem in its general form, allowing at most h bars to be installed at each step. Section 3.1 explains the finite element method for evaluating the structural deformation of an incomplete substructure. Section 3.2 introduces a graph-based representation for measuring the cost of an assembly sequence and formulates our sequence planning problem as a shortest-path problem using this graph.

3.1 Structural Analysis

We use the linear finite element method from [McGuire et al. 2000] to compute the quasi-static, elastic deformation of a frame structure during assembly. A brief introduction to this analysis method is given to make our paper self-contained. More details about this analysis method can be found in our supplementary.

Assuming linear elasticity, the deformed shape of an elastic bar b is determined by a deformation vector $\boldsymbol{u}_b$ describing the deformations of its two endpoints; see Figure 3(a). The deformation of each endpoint has 6 DOFs, including 3 DOFs for translation and 3 DOFs for rotation. The deformed bar b generates internal elastic forces

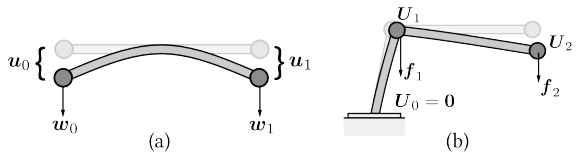

Fig. 3. (a) A deformed bar with a deformation vector $\boldsymbol{u} = [\boldsymbol{u}_0^T, \boldsymbol{u}_1^T]^T$ and an external force vector $\boldsymbol{w} = [\boldsymbol{w}_0^T, \boldsymbol{w}_1^T]^T$. (b) A deformed incomplete substructure, whose leftmost joint is anchored to the ground, so $\boldsymbol{U}_0 = \boldsymbol{0}$. Its global deformation vector is $\boldsymbol{U} = [\boldsymbol{U}_1^T, \boldsymbol{U}_2^T]^T$ and its global external force vector is $\boldsymbol{F} = [\boldsymbol{f}_1^T, \boldsymbol{f}_2^T]^T$.

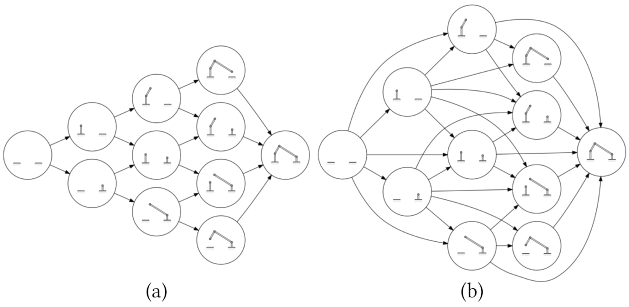

Fig. 4. (a) The state graph of a four-part assembly, when only one bar is installed at a time ($h = 1$). (b) The state graph of the same four-part assembly, when at most two bars can be installed at a time ($h = 2$). Both graphs share the same nodes, but the state graph in (b) has more edges than the state graph in (a).

$\boldsymbol{f}_b^{\text{int}}$ (i.e., stretching, bending and twisting).

$$\boldsymbol{f}_b^{\text{int}} = -\boldsymbol{K}_b \boldsymbol{u}_b \tag{1}$$

where $\boldsymbol{K}_b$ is the local stiffness matrix of the bar. The bar's self-weight is represented using a lumped point load $\boldsymbol{w}_b$, which acts on both ends of the bar; see Figure 3(a). The force equilibrium condition of the bar requires:

$$-\boldsymbol{K}_b \boldsymbol{u}_b + \boldsymbol{w}_b = 0 \tag{2}$$

When two bars are connected by a rigid joint c, their deformations at this joint are equal; see Figure 3(b). Thus, we define the system's degrees of freedom only at the joints. Each joint c is assigned a deformation vector $\boldsymbol{U}_c$ to represent the deformation of its associated bars at c. If a joint is fixed (e.g., anchored to the ground), its deformation vector must be zero. We stack all non-fixed joints' deformation vectors $\{\boldsymbol{U}_c\}$ to form a global deformation vector $\boldsymbol{U}$. The point load of a joint $\boldsymbol{f}_c$ is defined as the summation of its associated bars' loads at c. The global external force vector $\boldsymbol{F}$ is defined by stacking loads of all non-fixed joints $\{\boldsymbol{f}_c\}$. The force equilibrium condition of the frame system can be formulated as follows:

$$\boldsymbol{K}\boldsymbol{U} = \boldsymbol{F} \tag{3}$$

where $\boldsymbol{K}$ is the global stiffness matrix that is assembled from the local stiffness matrices $\{\boldsymbol{K}_b\}$.

We choose to use structural compliance to measure the structural deformation of an intermediate assembly state. Compliance is a widely-used quantity for measuring structural stiffness in topology optimization [Bendsoe and Sigmund 2003], which is formulated as:

$$C = \frac{1}{2}\boldsymbol{F}^T\boldsymbol{U} \tag{4}$$

A structure with a large compliance value generally has a large structural deformation. Therefore, our goal is to find the optimal assembly sequence whose intermediate substructures have the minimum amount of compliance.

3.2 State Graph

We adopt a graph representation for the sequence planning problem that allows for at most h bars to be installed at each step. Our state graph $G(V, E)$ is a directed graph whose nodes V represent all intermediate substructures that can be physically assembled. Such substructures are drawn in the circles representing the nodes V in Figure 4. Each node $v_i \in V$ stores two entries: 1) the indices of bars that have been installed; 2) the compliance of the intermediate substructure $C(v_i)$. Our state graph discards nodes that contain floating bars. Two graph nodes v_i and v_j are connected by an edge $e_{i,j}$ if v_j can be constructed from v_i by adding at most h bars. Figure 4(a) shows the state graph for $h = 1$. Installing more bars per step will significantly increase the total number of graph edges, even though it does not increase the total number of graph nodes, as shown in Figure 4(b) for the same example structure but with $h = 2$.

A path $P = \{v_{P_0}, v_{P_1}, \cdots\}$ from the start node (i.e., the empty substructure) to the final node (i.e., the completed structure) defines a *temporal coherent* assembly sequence, where partial structures present in a given state are also contained in ensuing states. The plot in Figure 1 provides a quantitative comparison between different assembly sequences, which visualizes the intermediate assembly states' compliances over the assembly progression. In practice, we need to define a single value metric for evaluating the assembly cost of a given assembly sequence. In this work, we choose to use the summation of the incomplete assembly states' compliances:

$$C(P) = \sum_i C(v_{P_i}) \tag{5}$$

Our sequence problem can then be formulated as a shortest-path problem. The optimal assembly sequence is the shortest path between the start and final node using the cost function defined by Equation 5. The famous Dijkstra's algorithm [Dijkstra 1959] can find the global minimum of the problem in $O(|E| + |V| \log |V|)$. However, the state graph of a medium-size frame structure (e.g., $n \geq 40$) is already too large to be handled even by the fastest shortest path algorithms. Theoretically, the number of graph nodes in our state graph is 2^n for a n-bar frame structure. Even after excluding infeasible nodes, solving the problem is still beyond the capability of most desktop computers. Besides, the number of successors of a graph node grows exponentially with the maximum step size h. Many search-based methods (e.g., greedy search) need to explore the successors of the current node, which becomes extremely inefficient for a large h.

4 TEMPORAL COHERENT TOPOLOGY OPTIMIZATION

In this section, we propose a new optimization-based sequence planning method inspired by topology optimization. In Section 4.1 we recall the basics of topology optimization and present a new

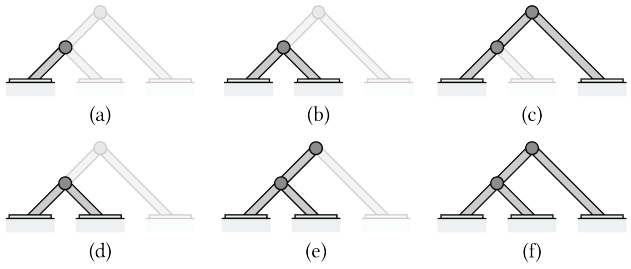

Fig. 5. Each of the substructures in (a), (b), and (c) has the lowest compliance among other substructures with the same number of bars. However, there is a bar in (b) that is not in (c). Directly linking (a), (b), (c), and (f) does not form a temporal coherent assembly sequence. A globally optimal assembly sequence is defined by substructures (a), (d), (e), and (f).

temporal coherent topology optimization framework for solving the single-bar-at-a-time sequence planning problem. In Section 4.2 we extend our framework to handle the installation of multiple bars per step.

4.1 Single-bar-at-a-time Sequence Planning

By definition, every assembly sequence in the single-bar-at-a-time setting is a permutation of the set $\{1, \cdots, n\}$. Exhaustively examining all sequences is impossible for large n because of its $O(n!)$ complexity. Alternatively, as described in Section 3.2, every assembly sequence can be viewed as a set of intermediate assembly states $\{A_t\}_{1 \le t \le n}$. Since only one bar is installed at a time, each substructure A_t must contain exactly t bars, and adjacent substructures must be *temporal coherent* (i.e., $A_t \subset A_{t+1}$). The minimum assembly cost E of constructing a n-bar frame structure is:

$$\begin{aligned} E = \min_{\{A_t\}} \quad & \sum_{t=1}^{n} C(A_t) \\ \text{s.t.} \quad & |A_t| = t, \\ & A_t \subset A_{t+1} \end{aligned} \tag{6}$$

where $C(A_t)$ is the compliance of the incomplete substructure A_t and $|\cdot|$ measures the number of bars in A_t.

Applying this new formulation still requires solving a complex discrete problem. To make this formulation more manageable, we first exclude the temporal coherent constraints $A_t \subset A_{t+1}$ in Equation 6, transforming it into multiple separate optimization problems.

$$\begin{aligned} A_t^* = \arg\min_{A_t} \quad & C(A_t) \\ \text{s.t.} \quad & |A_t| = t \end{aligned} \tag{7}$$

where A_t^* is the stiffest substructure with exactly t bars.

Suppose that the set $\{A_1^*, \cdots, A_n^*\}$ forms a temporal coherent assembly sequence (i.e., $A_t^* \subset A_{t+1}^*$), the temporal coherent constraints in Equation 6 holds and the optimal assembly sequence is found. Thus, one of the key issues is solving the sub-problem defined in Equation 7.

Finding the stiffest substructure using a subset of bars is a classic problem in topology optimization [Stolpe 2016]. We denote the presence of the bar b_i in the substructure A as a binary variable ρ_i. The bar b_i is installed when $\rho_i = 1$. The variable ρ_i can be viewed as the factitious thickness or pseudo-density of b_i. The material stiffness (i.e., Young's modulus) and self-weight of b_i is then a function of ρ_i:

$$\boldsymbol{K}_i(\rho_i) = (\rho_i + \varepsilon)\boldsymbol{K}_i \tag{8}$$

$$\boldsymbol{w}_i(\rho_i) = \rho_i \boldsymbol{w}_i \tag{9}$$

where ε is a very small number to prevent singularity. In this work, we favor our linear interpolation law over the conventional SIMP method [Bruyneel and Duysinx 2005] to guarantee the convexity of our relaxed formulation. The global displacement vector $\boldsymbol{U}(\boldsymbol{\rho})$ and the compliance $C(\boldsymbol{\rho})$ are also functions of the global thickness vector $\boldsymbol{\rho}$.

$$\boldsymbol{U}(\boldsymbol{\rho}) = \boldsymbol{K}^{-1}(\boldsymbol{\rho})\boldsymbol{F}(\boldsymbol{\rho}) \tag{10}$$

$$C(\boldsymbol{\rho}) = \frac{1}{2}\boldsymbol{F}(\boldsymbol{\rho})^T\boldsymbol{K}^{-1}(\boldsymbol{\rho})\boldsymbol{F}(\boldsymbol{\rho}) \tag{11}$$

The sub-problem of finding the stiffest substructure A_t^* in Equation 7 can be reformulated as a topology optimization problem with binary constraints imposed on $\boldsymbol{\rho}$:

$$\begin{aligned} \min_{\boldsymbol{\rho}} \quad & \frac{1}{2}\boldsymbol{F}(\boldsymbol{\rho})^T\boldsymbol{K}^{-1}(\boldsymbol{\rho})\boldsymbol{F}(\boldsymbol{\rho}) \\ \text{s.t.} \quad & \sum_i \rho_i = t, \quad \rho_i \in \{0, 1\} \end{aligned} \tag{12}$$

Figure 5 shows the results of finding the stiffest substructures with 1, 2, and 3 bars in a four-bar assembly. Ideally, connecting these substructures should form a feasible assembly sequence. However, without the temporal coherent constraints in Equation 6, the resulting assembly sequence could be infeasible. For instance, there exists a bar installed in Figure 5(b), which later disappears in Figure 5(c). To obtain feasible assembly sequences, we propose a new temporal coherent topology optimization that simultaneously solves a series of topology optimizations with temporal coherent constraints:

$$\begin{aligned} \min_{\{\boldsymbol{\rho}^t\}} \quad & \frac{1}{2}\sum_{1 \le t \le n} \boldsymbol{F}(\boldsymbol{\rho}^t)^T\boldsymbol{K}^{-1}(\boldsymbol{\rho}^t)\boldsymbol{F}(\boldsymbol{\rho}^t) \\ \text{s.t.} \quad & \sum_i \rho_i^t = t, \quad \rho_i^t \in \{0, 1\}, \\ & \rho_i^t \le \rho_i^{t+1} \end{aligned} \tag{13}$$

where ρ_i^t is the factitious thickness of bar b_i in the sub-assembly A_t. The temporal coherent constraint $\rho_i^t \le \rho_i^{t+1}$ ensures that the bars which appear at time step t cannot disappear at subsequent steps.

4.2 Multiple-bars-at-a-time Sequence Planning

Our temporal coherent topology optimization can be extended to solve multiple-bars-at-a-time sequence planning problems. The primary incentive behind this is that enabling the concurrent assembly of several bars could lower the compliance of intermediate substructures, which can be supported through the use of multiple robotic systems or pre-assembled modules.

A straightforward way of updating the material budget constraints in Equation 13 for assembling multiple bars per step is:

$$\sum_i \rho_i^t = ht, \quad 1 \le t \le \left\lfloor \frac{n}{h} \right\rfloor \tag{14}$$

Fig. 6. An assembly sequence computed using our BNB solver for constructing a 24-bar TOWER model [Fox and Schmit Jr 1966]. The sequence contains 6 assembly stages with non-uniform step sizes, where at most 5 bars are installed at a time. The right plot shows the assembly cost of the incumbent solution and the current best lower bound over the computational time. A global minimum is found when these two lines meet.

This formulation only supports assembly processes with a uniform step size in which the number of bars installed at each step is constant. However, in some scenarios, assembly sequences with non-uniform step sizes can have a lower assembly cost. To adaptively select the number of bars installed at each step, we update the material budget constraints in our sequence topology optimization framework as follows:

$$\begin{aligned} \min_{\{\boldsymbol{\rho}^t\}} \quad & \frac{1}{2} \sum_{1 \le t \le M} \boldsymbol{F}(\boldsymbol{\rho}^t)^T \boldsymbol{K}^{-1}(\boldsymbol{\rho}^t) \boldsymbol{F}(\boldsymbol{\rho}^t) \\ \text{s.t.} \quad & 0 \le \sum_i \rho_i^{t+1} - \sum_i \rho_i^t \le h, \\ & \sum_i \rho_i^0 = 0, \quad \sum_i \rho_i^M = n, \\ & \rho_i^t \in \{0,1\}, \\ & \rho_i^t \le \rho_i^{t+1} \end{aligned} \tag{15}$$

where we restrict the number of bars added at each step to be smaller and equal to the maximum step size h. Here, M is a predefined maximum number of total assembly steps, including the final step. Setting a large assembly step M does not exclude potential solutions, as our framework accommodates skipping steps (i.e., $\rho_i^t = \rho_i^{t+1}$).

5 SOLVERS

Solving our mixed integer program described in Equation 15 is a challenging combinatorial problem. Section 5.1 presents the holistic approach we use for solving mixed integer optimization. While, theoretically, this method should be able to fulfill our requirements, in practice, we find that even the state-of-the-art holistic solver still cannot efficiently handle some cases. Therefore, we propose a hierarchical decomposition approach based on the divide-and-conquer method to reduce computational time.

5.1 Holistic Approach

The goal of our holistic approach is to incorporate all assembly stages concurrently within a single, large-scale optimization process. A standard way to solve mixed integer programs is to relax the binary constraint $\rho_i \in \{0,1\}$, replace it with a continuous constraint $\rho_i \in [0,1]$, and use a Branch-and-Bound (BnB) method to solve the relaxed problem [Gupta and Ravindran 1985]. Conceptually, a BnB method uses a binary tree search to recursively split the search space by setting one of the $\{\rho_i^t\}$ to be zero or one. At each tree node, the BnB method solves a continuous optimization which is derived from the relaxed problem to estimate a lower bound on the binary problem's objective value. The BnB algorithm can assert that it converges to global optimality when the gap between the incumbent solution's objective and the current best lower bound are within a user-specified threshold. As proven in the supplementary material, the relaxed problem of Equation 15 is convex and can always be solved to a global minimum for an accurate lower-bound estimation. Figure 6 shows the evolution of the lower-bound and incumbent solution's cost, the current best-found solution's cost, by running the BnB solver on an exemplary problem to obtain the globally optimal sequence. In many of our experiments, the gap generally does not converge to zero, yet the distance to the lower bound can help users decide whether continuing the optimization is worthwhile. In this work, we use the BnB implementation provided in the commercial optimization solver Knitro [Byrd et al. 2006]. In the followings, we use HOLISTIC to refer to our holistic approach that employs the branch-and-bound solver from Knitro.

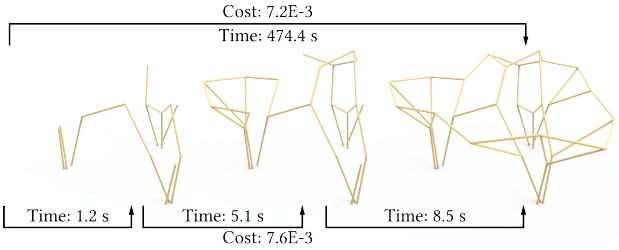

Fig. 7. An illustration of how the assembly sequence of a 45-bar roboarch model [2018] is computed using our hierarchical decomposition approach. At most 4 bars can be installed at a time. Directly calculating the assembly plan using our holistic approach takes 474.4 s. Our hierarchical decomposition approach first finds two temporal coherent landmarks with 15 and 30 bars, which takes 3.0 s. Then, each sub-problem is solved using the holistic approach with given starting and ending intermediate substructures. Our hierarchical decomposition approach takes 17.8 s, which is 26 times faster than our holistic approach and only increases the assembly cost by 5.5%.

5.2 Hierarchical Decomposition Approach

The efficiency of our HOLISTIC solver depends on the number of binary variables. For a medium size model ($n = 40$), having a small step size (e.g., $h = 1$) requires a large number of binary variables (e.g., n^2), which cannot be efficiently handled by our HOLISTIC solver. To accelerate the computation, we propose another solver that first

finds a given number (e.g. z) of sub-assemblies $\{L_k\}_{1\le k\le z}$ that are temporary coherent, and then enforce our assembly sequence to go through these sub-assemblies. These sub-assemblies are named *landmarks*, which should satisfy $L_k \subset L_{k+1}$. These landmarks decompose the original sequence planning problem into disjointed sub-problems, each of which only needs to find the optimal assembly sequence starting at the landmark L_k and ending at the landmark L_{k+1}. Because the number of undecided bars in each sub-problem is significantly reduced, these sub-problems should be solved using our HOLISTIC solver. However, if a sub-problem is still too difficult to be solved, our method can compute new landmarks to decompose this sub-problem further. Our method terminates if all sub-problems are solved. Similar to other sequence planning works that use a divide-and-conquer strategy [Deuss et al. 2014; Huang et al. 2016], this landmark strategy cannot guarantee global optimality but can often generate good results significantly faster.

In this work, we use our temporal coherent topology optimization to compute the landmarks to evenly partition the frame structure into z landmarks using Equation 13 with the constraints of Equation 14. The number of bars in landmark L_k is at most $\lceil (nk)/(z+1) \rceil$. In practice, to further accelerate the computation, we can solve this landmark problem in a bottom-up fashion. Starting from only solving for L_1, we iteratively computes the landmark L_k by fixing all previous landmarks $\{L_1, \cdots, L_{k-1}\}$ and neglecting the assembly costs of following, uncomputed landmarks $\{L_{k+1}, \cdots, L_z\}$. In this way, because we only consider decision variables $\boldsymbol{\rho}^k$ at each iteration k instead of $\{\boldsymbol{\rho}^t\}_{1\le t\le z}$, the optimization problem is much smaller and thus faster to solve. We find that in practice using this iterative strategy can effectively find good landmarks. We name this hierarchical decomposition strategy the z-LANDMARK method. A sequence computed using this strategy can be found in Figure 7.

6 ASSEMBLY WITH SUPPORTS

Increasing the number of bars h installed at each step potentially reduces the assembly cost in terms of compliance, as unstable substructures can be avoided. However, h is usually constrained by the available resources, e.g. workers or robots. We propose to use supports to fix the bars' positions, in order to further reduce the assembly cost without the need to increase h. Supports can be classified into two types: static scaffolding (Section 6.1) and dynamic holding (Section 6.2). The bars fixed by static scaffolding are permanent, i.e. not removable until the end of the construction, while dynamic holding allows for dynamic relocation of a fixed number of static supports. We extend our temporal coherent topology optimization framework to handle the supporting operations and minimize the total assembly cost by using a given number of supports S.

6.1 Assembly with Static Scaffolding

Most real-world construction projects make use of a dense scaffold to support intermediate substructures, which is difficult and time-consuming to build. We propose a new sequence planning problem that, at most, fixes S ($S << n$) bars using static scaffolding to reduce the assembly cost. Solving this new planning problem involves simultaneously planning for the assembly sequence and the bars

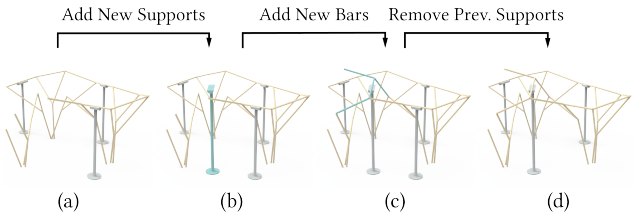

Fig. 8. New supports are first installed in (b) before the new bars are installed in (c). Supports that are no longer needed are removed in (d). Newly introduced bars or supports are highlighted in cyan.

that need to be fixed. We tackle this problem by extending our method to include the "fix" operation.

A bar is fixed if the deformation vectors of its two ends are zero. We denote $\boldsymbol{U}_{b_i}$ as the deformation vector of the i-th bar, a sub-vector of the global deformation vector $\boldsymbol{U}$. Fixing the i-th bar is equivalent to setting $\boldsymbol{U}_{b_i} = \boldsymbol{0}$. We introduce a new set of decision variables $\{s_i\}$ to describe the fixation status of the bars:

$$s_i = 1 \Rightarrow \boldsymbol{U}_{b_i} = \boldsymbol{0} \tag{16}$$

However, this conditional constraint makes the problem difficult to solve using our formulation: We cannot treat s_i in the same way as ρ_i in Section 4.1, as a partially fixed bar is not well defined. We instead formulate Equation 16 as an additional, conditional stiffness matrix $s_i \boldsymbol{P}_i$, which is added to the global stiffness matrix $\boldsymbol{K}$. Each $\boldsymbol{P}_i$ is a constant matrix:

$$(\boldsymbol{P}_i)_{j,k} = \begin{cases} \mu & j = k \text{ and } k \in b_i \\ 0 & \text{others} \end{cases} \tag{17}$$

We choose μ to be a relatively large number depending on the material (about 10^3 times the material density). This enforces that the condition in Equation 16 is satisfied. The new global stiffness matrix can then be written as:

$$\boldsymbol{K}^*(\boldsymbol{\rho}^t, \boldsymbol{s}) = \boldsymbol{K}(\boldsymbol{\rho}^t) + \sum_i s_i \boldsymbol{P}_i \tag{18}$$

By updating the global stiffness matrix, our temporal coherent topology optimization can be extended to include static scaffolding:

$$\begin{aligned} \min_{\boldsymbol{s},\{\boldsymbol{\rho}^t\}} \quad & \sum_{1\le t\le M} \frac{1}{2} \boldsymbol{F}(\boldsymbol{\rho}^t)^T \left(\boldsymbol{K}^*\right)^{-1} \boldsymbol{F}(\boldsymbol{\rho}^t) \\ \text{s.t.} \quad & 0 \le \sum_i \rho_i^{t+1} - \sum_i \rho_i^t \le h, \\ & \sum_i \rho_i^0 = 0, \quad \sum_i \rho_i^M = n, \quad \sum_i s_i = S, \\ & \rho_i^t \le \rho_i^{t+1}, \\ & s_i, \rho_i^t \in \{0, 1\} \end{aligned} \tag{19}$$

6.2 Assembly with Dynamic Holding

A dynamic holding formulation is useful if supports can easily be moved during assembly (e.g., when robots are used to hold the bars). We can extend Equation 19 to include dynamic holding by defining

Table 1. A comparison of our methods to the baseline methods in the single-bar-at-a-time setting on our 100-model dataset. The second row specifies the model sizes clustered as small ($n < 40$), medium ($40 \leq n < 60$), and large ($60 \leq n \leq 150$). All assembly costs are normalized by taking the results from the FORWARD-GREEDY method as a reference. Every entry in the table shows three values in the following format: average (minimum and maximum). With many variables, our HOLISTIC approach often fails to converge, leading to suboptimal performance compared to the 2-LANDMARK approach in most test cases. More detailed comparisons between our HOLISTIC and 2-LANDMARK approach can be founded in our supplementary material.

h	Methods	Assembly Cost			Time (s)		
		Small	Medium	Large	Small	Medium	Large
1	Forward-greedy	1	1	1	0.02 (0.01~0.04)	0.06 (0.03~0.11)	0.32 (0.10~1.5)
	Backward-greedy	1.88 (0.79~6.88)	2.16 (0.49~14.75)	1.87 (0.47~8.54)	0.05 (0.02~0.08)	0.14 (0.07~0.32)	0.84 (0.23~3.5)
	Backtrack-greedy	0.96 (0.84~1.00)	0.97 (0.84~1.00)	0.99 (0.70~1.00)	10	10	10
	2-Landmark (Ours)	0.80 (0.50~1.04)	0.68 (0.36~1.07)	0.72 (0.34~1.02)	44.9 (0.73~142)	113 (2.2~686)	108 (6.4~1030)
	Holistic (Ours)	0.97 (0.66~1.54)	-	-	1274 (2.9~3526)	-	-

a set of decision variables $\{s_i^t\}$ for each assembly step:

$$\begin{aligned} \min_{\{\boldsymbol{s}^t\},\{\boldsymbol{\rho}^t\}} \quad & \sum_{1\leq t\leq M} \frac{1}{2}\boldsymbol{F}(\boldsymbol{\rho}^t)^T\left(\boldsymbol{K}^*\right)^{-1}\boldsymbol{F}(\boldsymbol{\rho}^t) \\ \text{s.t.} \quad & 0 \leq \sum_i \rho_i^{t+1} - \sum_i \rho_i^t \leq h, \\ & \sum_i \rho_i^0 = 0, \quad \sum_i \rho_i^M = n, \quad \sum_i s_i^t = S, \\ & \rho_i^t \leq \rho_i^{t+1}, \quad s_i^t \leq \rho_i^t, \\ & s_i^t, \rho_i^t \in \{0,1\} \end{aligned} \tag{20}$$

The constraint $s_i^t \leq \rho_i^t$ ensures that only installed bars can be fixed using dynamic holding. With the same h (i.e., the maximum step size) and M (i.e., the number of assembly steps), the number of variables here is doubled compared to Equation 15. Because of the large number of design variables, we prefer to solve this optimization using our z-LANDMARK solver described in Section 5.2. In practice, supports used in dynamic holding cannot be transferred instantly between assembly steps. We run a post-processing algorithm to compute additional steps for transition, which might temporarily need more supports than the given number of supports S; see Figure 8.

7 RESULTS AND DISCUSSION

We implement our method in C++ and use Knitro 13.1 [Byrd et al. 2006] for solving our mixed integer optimization. We run our optimization on a Linux workstation equipped with an AMD Threadripper Pro 5995WX (64 Cores) and 128 GB of memory.

Dataset. A frame structure dataset is created to evaluate our method's effectiveness compared with existing approaches. Collections of freeform frame structures are difficult to find. As a result, we create a new frame structure dataset derived from the Thingi10k dataset [Zhou and Jacobson 2016]. We first manually select 100 models from the Thingi10k dataset and then simplify them using an edge-collapse algorithm to reduce their total number of edges. We then extract the wireframes from the simplified meshes to form new frame structures. Every frame structure is scaled to fit into a 3m unit box. We set a uniform bar radius (R = 5mm), density (D = 5.8 kN/m^3), Young's modulus (E = 12.9 GPa), and shear modulus (G = 4.8 GPa) for all frame structures in our dataset. Note that our method can support frame structures composed of bars with non-uniform materials and radii. In summary, our dataset contains 100 frame structures of which 21 are small-size models ($n < 40$), 53 medium-size models ($40 \leq n < 60$), and 26 large-size models ($60 \leq n \leq 150$). Please refer to the supplementary materials for a visual summary of these structures.

7.1 Single-bar-at-a-time Comparisons

We compare our algorithm with different baseline methods in the single-bar-at-a-time setting. The baseline algorithms we use are search-based methods, namely the FORWARD-GREEDY, BACKWARD-GREEDY and BACKTRACK-GREEDY [McEvoy et al. 2014]. The FORWARD-GREEDY method keeps adding bars to an initially empty structure, while the BACKWARD-GREEDY method keeps disassembling bars from the initially complete structure to improve the assembly cost. The FORWARD- and BACKWARD-GREEDY methods terminate after all bars have been assembled or disassembled, respectively. The BACKTRACK-GREEDY method is derived from the FORWARD-GREEDY method, but keeps exploring new assembly sequences using backtracking until a time limit (i.e., 10 s) is reached. Table 1 shows a comparison of our method with the baseline algorithms regarding the assembly cost and computational time.

Forward-greedy. Our teaser figure (Figure 1) illustrates an example where our method produces a solution with lower assembly cost than the FORWARD-GREEDY method. Our approach can plan in the long-term and is able to take more costly steps at the beginning in return for a reduction in assembly cost later. Greedy-based methods are typically short-sighted and do not have this ability.

Backward-greedy. Even though the BACKWARD-GREEDY method on average doubles the assembly cost compared to the FORWARD-GREEDY method, in the best case it can sometimes find assembly sequences with only half the cost of FORWARD-GREEDY.

Backtrack-greedy. Even though this method benefits from a significant increase in search time, it only manages to reduce the assembly costs by 2.82% over the FORWARD-GREEDY method. The reason is that problematic search choices are often made during the early stages of the search, which is difficult to correct using backtracking.

We generally cannot directly use our HOLISTIC solver for models with more than 40 bars in the single-bar-at-a-time setting, which convergent slowly. We mainly compare the baseline methods with

Table 2. Results of our benchmark in the multiple-bars-at-a-time setting on our 100-model dataset. The first column shows the maximum step size h. The table entries are marked with "-" if running the corresponding algorithm takes more than 1000s. Both HOLISTIC and z-LANDMARK are our proposed methods. Our HOLISTIC approach may not always converge, leading to suboptimal results when compared to our Z-LANDMARK approach. We observe that our HOLISTIC approach can obtain assembly costs that are lower or the same as the costs obtained from our z-LANDMARK approach on a considerable percentage of test models (i.e., 42.9%, 32.4%, and 82.4% for $h = 4, 6$, and 10). Please refer to our supplementary material for more detailed comparisons.

h	Methods	Assembly Cost			Time (s)		
		Small	Medium	Large	Small	Medium	Large
4	Merge	1	1	1	0.03 (0.01~0.05)	0.08 (0.04~0.17)	0.51 (0.13~2.2)
	Greedy	0.75 (0.25~0.98)	0.70 (0.34~0.93)	0.78 (0.25~1.00)	0.71 (0.12~1.4)	2.3 (0.31~12.2)	104 (1.8~1282)
	z-Landmark (Ours)	0.66 (0.16~0.98)	0.59 (0.30~0.87)	0.64 (0.19~1.03)	13.0 (0.41~42.1)	108 (2.7~556)	244 (21.2~708)
	Holistic (Ours)	0.70 (0.17~0.98)	-	-	339 (1.9~913)	-	-
6	Merge	1	1	1	0.03 (0.01~0.05)	0.08 (0.04~0.17)	0.51 (0.13~2.2)
	Greedy	0.76 (0.24~0.99)	0.64 (0.33~1.00)	-	25.5 (2.3~52.7)	136 (8.0~1164)	-
	z-Landmark (Ours)	0.69 (0.23~0.99)	0.57 (0.30~0.94)	0.62 (0.19~0.97)	4.2 (0.87~13.6)	32.5 (2.8~142)	97.2 (5.3~315)
	Holistic (Ours)	0.70 (0.23~0.99)	0.68 (0.31~1.10)	-	187 (2.8~906)	480 (23.2~988)	-
10	Merge	1	1	1	0.03 (0.01~0.05)	0.08 (0.04~0.17)	0.52 (0.13~2.2)
	z-Landmark (Ours)	0.69 (0.19~0.97)	0.61 (0.30~0.98)	0.64 (0.29~1.04)	18.4 (0.06~75.9)	10.9 (0.54~96.0)	78.4 (4.1~181)
	Holistic (Ours)	0.69 (0.19~0.97)	0.63 (0.35~1.22)	-	42.8 (0.05~570)	324 (0.89~964)	-

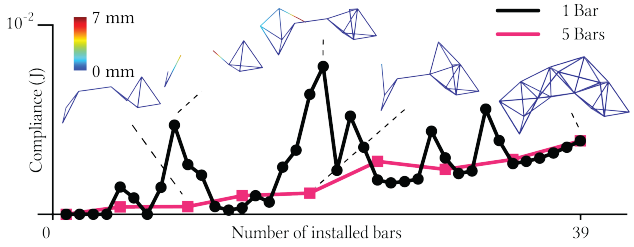

Fig. 9. Installing a 39-part bridge structure [2021] by assembling at most 1 (black) or 5 (magenta) bars per step. The plot shows the compliance of the two sequences over the number of installed bars. The 1-Bar sequence is computed using our z-LANDMARK solver, and the 5-Bar sequence is computed using our HOLISTIC solver. All bars have the same material properties $R = 10$ mm, $D = 12.7$ kN/m^3, $E = 2.07$ GPa, $G = 0.89$ GPa.

our 2-LANDMARK solver, which recursively finds 2 temporal coherent landmarks to decompose the sequence planning problem until each sub-problem can be efficiently resolved using our HOLISTIC approach (i.e., the number of undecided bars is less than 15). Our 2-LANDMARK method can reduce the assembly cost by 28% over the FORWARD-GREEDY approach.

7.2 Multiple-bars-at-a-time Comparisons

Simultaneously assembling multiple bars can reduce the structural deformation of the intermediate substructures as opposed to the single-bar-at-a-time assembly process; see Figure 9. Previous sequence planning methods [Huang et al. 2021; McEvoy et al. 2014] only compute assembly plans that install one bar at a time. We have to extend their methods to handle multiple-bars-at-a-time assembly for comparison. A simple extension is to combine h consecutive assembly steps of their single-bar-at-a-time sequences into one step. We call this the MERGE method. An alternative baseline approach involves extending the GREEDY method to accommodate multi-bar assembly by examining all combinations of installing at most h bars during each search iteration; see our supplementary material.

The MERGE method can handle arbitrary h but often generates sequences with high assembly costs (Figure 10). This happens because the merged sequence can include intermediate assembly states with high assembly costs from the unmerged sequences.

The GREEDY method performs well on medium-size models when h ranges from 2 to 4. However, it struggles to efficiently address the sequencing issue for larger step sizes (e.g., $h \geq 6$), as the number of successors for the current search node increases exponentially with h. Evaluating the assembly costs of these successor nodes can take thousands of seconds.

Our z-LANDMARK method performs better than baseline methods in various model sizes n and maximum step sizes h. Statistics of the benchmark are shown in Table 2. In general, our HOLISTIC solver can efficiently compute assembly sequences on small-size models $n < 40$ with $h \geq 4$ and on medium-size models $n < 60$ with $h \geq 6$. We set a time limit of 1000s for our HOLISTIC solver and report the time at which it generates the current best results. In terms of implementation, Knitro has three types of heuristics (i.e., "Basic", "Advanced", and "Extensive") for solving mixed integer problems. In our experiment, we favor the "Advanced" heuristic, which exhibits the most consistent performance. More comparisons between different heuristics can be found in the supplementary.

The large-size models ($n \geq 60$) currently can only be efficiently handled by our z-LANDMARK solver. During our experiments, we set our z-LANDMARK solver to divide the whole assembly problem into sub-problems that have at most 30, 40, and 40 uninstalled bars when $h = 4, 6$, and 10. Each sub-problem is solved using our HOLISTIC approach. For $h \geq 6$, our z-LANDMARK method is significantly faster than the GREEDY and has less assembly cost than the MERGE method. Currently, our z-LANDMARK method still has a limitation in selecting the landmarks, because sometimes the optimal landmarks might not

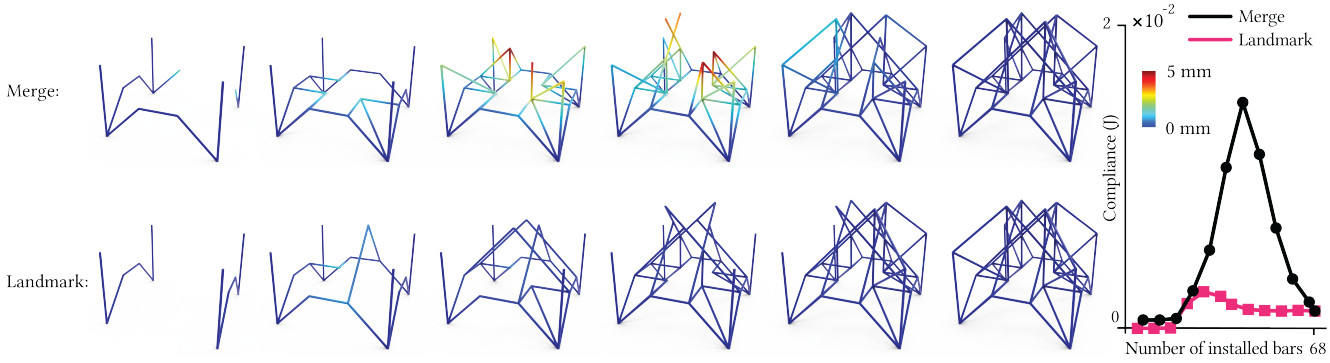

Fig. 10. Assembly sequences for constructing a 68-part `frustum` structure from [Huang et al. 2021] by installing at most 6 bars per step, color-coded by blue to yellow representing small to large deformation. The sequence generated by the MERGE method is shown on top, and the sequence generated by our 2-LANDMARK solver is shown on the bottom. Each sub-problem generated by our 2-LANDMARK solver is solved using our HOLISTIC approach. The plot shows the compliance of the two sequences over the number of installed bars. Our method reduces the assembly cost by 77% and takes 122.6s to compute. All bars have the same material properties $R = 10$ mm, $D = 12.7$ kN/m^3, $E = 2.07$ GPa, $G = 0.89$ GPa.

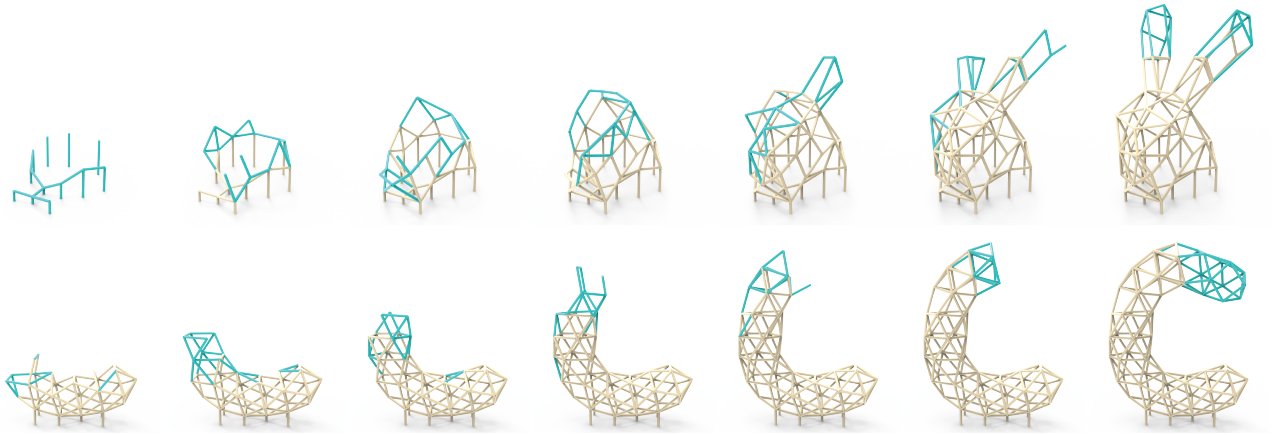

Fig. 11. Layer decomposition results computed by our z-LANDMARK solver. Both models are from [Huang et al. 2016]. (top) The 141-bar bunny model is decomposed into 7 layers, each of which has 20 bars. Computing the sequence for this bunny model takes our method 8.4s and ends up with an assembly cost of 8.8×10^{-2} (while [Huang et al. 2016] have an assembly cost of 11.2×10^{-2}). (bottom) The 199-bar C model is decomposed into 10 layers, each of which has the same number of bars as [Huang et al. 2016]. To compute this C model, our method takes 84.4s and has an assembly cost of 4.04×10^{-2} (compared to [Huang et al. 2016] 4.19×10^{-2}). All bars have the same material properties $R = 20$ mm, $D = 12.7$ kN/m^3, $E = 2.07$ GPa, $G = 0.89$ GPa.

be part of the optimal assembly sequence, as illustrated in Figure 14. In practice, we prefer using our HOLISTIC solver as long as it can solve the problem and we are under no time constraints.

Our z-LANDMARK solver can also be applied to solve the layer decomposition problem in [Huang et al. 2016]; see Figure 11. Our method can not only find a layer decomposition where each layer has a non-uniform number of bars as in [Huang et al. 2016], but with lower compliance cost. In addition, it can also uniformly partition the model into a given number of layers if desired by the users.

7.3 Extensions

Static scaffold. We test our method for assembling the ARCH model from [Bruun et al. 2022] using static scaffolding (Figure 12), where three supporting poles are used, and four bars can be assembled per step. We solve our optimization in Equation 19 using our HOLISTIC solver, which takes 1642s and reduces 98% of the cost compared to the original sequence without scaffolding.

Dynamic holding. Figure 12 visualizes the assembly sequence of building the MYCOTREE structure from [Lee 2018] using dynamic holding. Because of the large model size, we solve our optimization in Equation 20 using our 4-LANDMARK solver, which takes 2130s. Compared to the four-bars-at-a-time sequence without supports, deploying three dynamic holdings can effectively reduce 96% of the assembly cost.

MR-assisted Assembly. Based on the sequence output by our method, it is easy to generate visual assembly instructions for one or multiple users. Heads-up displays (HUDs) or Augmented/Mixed Reality

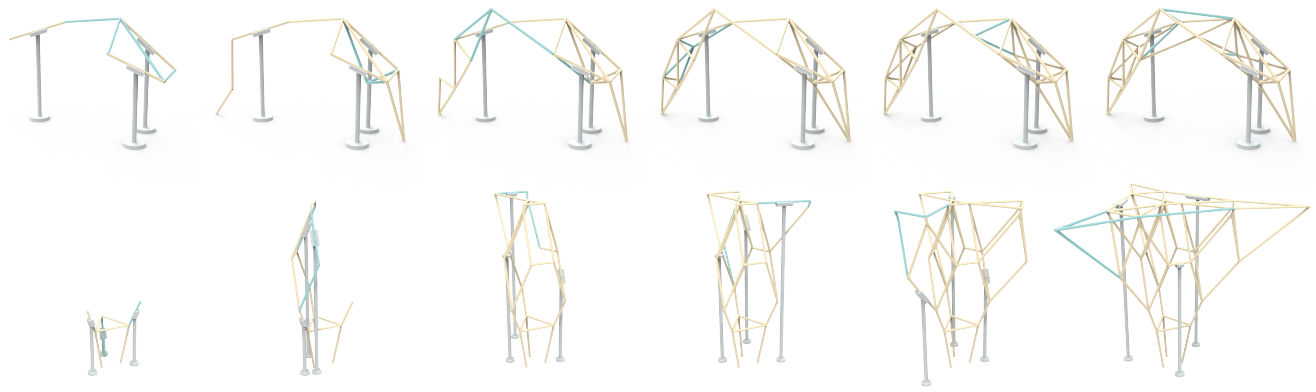

Fig. 12. The first row visualizes the assembly sequence of the ARCH model from [Bruun et al. 2022] by using 3 static supports and assembling 4 bars per step. The second row visualizes the assembly sequence of the MycoTree model from [Lee 2018] using 3 dynamic holdings and assembling 4 bars per step. All bars share the same material properties $R = 10$ mm, $D = 5.8$ kN/m^3, $E = 12.9$ GPa, $G = 4.6$ GPa.

(AR/MR) glasses [Jahn et al. 2020; Mitterberger et al. 2020] are well suited to display these instructions in the three-dimensional space. We implemented a prototype of such an assistive system on the Microsoft Hololens 2 MR glasses. The application that runs on the glasses was developed in the Unity game engine and uses a QR-code tracking system to align the user's frame of reference with the desired target pose of the structure. The user can interact with virtual buttons to run through the steps of the assembly. Thanks to the improved structural compliance using our method, the virtual and real structures have a better visual match compared to sub-optimal sequences. This improves the user's ability to orient and assemble the bars correctly without the need for any tracking or vision-based feedback. An example of the assembly process using the MR system can be found in the accompanying video material and in Figure 13.

Robot-assisted Assembly. An important aspect of digital fabrication is the ability for robots to automatically assemble the structure. There are many major technical challenges in robotic assembly that we do not claim to have solved in this paper, which includes grasp planning, trajectory optimization, and close-loop perception and control. However, a bad assembly sequence can lead to large structural deformations during the assembly that require the robotic system to correct for those offsets. A sequence with smaller deformation, therefore, is essential to the robotic assembly's success. To illustrate the potential for combining our sequence planning and multi-robot assembly planning, an optimized sequence computed by our algorithm is given to a trajectory optimization algorithm with collision avoidance for the multi-robot team, inspired by the method presented in [Zimmermann et al. 2020]. We show a sequence for the assembly of a bridge structure with four robots in the accompanying video material. A still image of the assembly can be seen in the inset.

8 CONCLUSION & FUTURE WORK

In this work, we propose an optimization-based framework that computes assembly sequences for constructing frame structures to minimize the structural deformation of their incomplete assembly states. We formulate our sequencing problem as a topology optimization with temporal coherent constraints and solve the problem using mixed integer optimization. Our method can effectively solve single-bar-at-a-time and multiple-bars-at-a-time sequence planning problems. Our method can be further extended to solve the assembly process, including static scaffolding and dynamic holdings. To illustrate potential application scenarios, we physically build a frame structure with the assistance of an augmented reality headset and demonstrate robotic assembly following our computed sequence in a simulated environment.

Limitations and future work. First, our optimization-based method can only efficiently handle input structures with less than 220 bars. Planning additional supports during assembly increases the computational time, limiting the maximum model size to 80 bars. Currently, we use Knitro to solve our optimization, which is a general-purpose MIP solver. Developing a customized MIP solver with dedicated branch-and-bound heuristics for our formulation to improve its computational efficiency would be a good research topic.

Second, our landmark solver can sometimes generate sequences with large assembly costs due to misleading landmarks (see Figure 14 for an example). However, because of the computing speed gain from our landmark strategy, experimenting with different layer sizes and comparing the sequence performance work well in practice.

Third, we use compliance to define the assembly cost of a given assembly sequence. Integrating other cost functions, such as the maximum displacement or buckling effect, into our optimization-based framework would be an exciting yet challenging future direction. Many of these cost functions are non-convex and thus more challenging to be solved using off-the-shelf MIP solvers.

Fourth, our method currently ignores collision aspect of the assembly agents (human or robot). The installed bars can block assembly agents from assembling new bars following the assembly

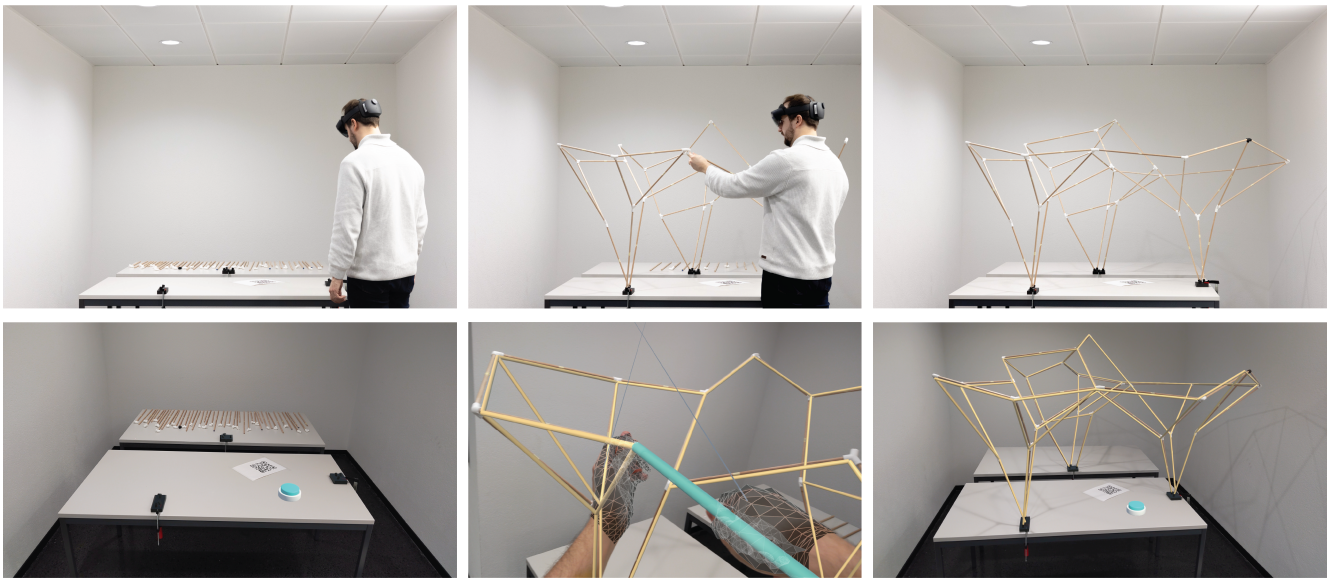

Fig. 13. The ROBOARCH structure is assembled by a single user who is guided by our MR system. The top row shows the assembly process. In the bottom row, we show the user interface on the headset, in which users can check which bars have been installed (yellow) and which bar will have to be installed next (cyan). The user needs to press a virtual button to move to the next assembly step. The final structure is almost perfectly aligned with the virtual model. Fabrication tolerances and structural deformations not captured by our model cause minor misalignment.

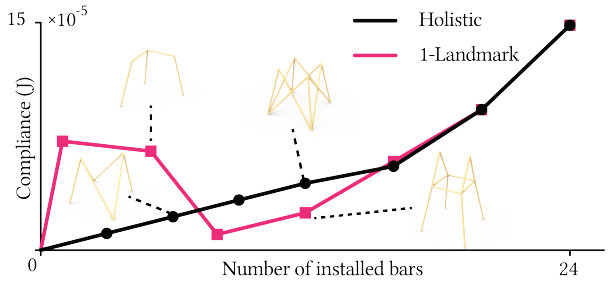

Fig. 14. The compliance-step curves of sequences generated by our 1-LANDMARK (magenta) and HOLISTIC solver (black) for assembling the TOWER model by installing 4 bars at a time. Because of its procedural nature, 1-LANDMARK solver selects a 12-bar landmark that is not part of the globally optimal sequence (the bottom right substructure). The sequence generated by our HOLISTIC solver has less overall assembly cost (4.0×10^{-4} J) than the sequence generated by our 1-LANDMARK SOLVER (4.7×10^{-4} J), despite the landmark shown here being the stiffest substructure with 12 bars.

sequence generated by our algorithm. Planning a sequence that considers both collision and structural deformation could be a promising future work, which could use the landmarks generated by our method to reduce search space.

Fifth, in our MR-assisted assembly prototype, we found that the deformations in the real structure exceeded the deformations predicted by our FEA model, due to the fabrication tolerances in the joints and small discrepancies between the model parameters and the material properties of the wooden beams we used. Improving our model through fine-tuning of the material parameters would be interesting for future work, especially for work that includes robotic assembly agents.

Lastly, we would like to extend our formulation to other assembly processes, such as building masonry shells using rigid blocks [Deuss et al. 2014] and tensegrity structures [Pietroni et al. 2017].

ACKNOWLEDGMENTS

We thank the reviewers for valuable comments, Huimin Shan for her help in fabrication, and Edvard Brunn, Alec Jacobson, and Caitlin Mueller for proving their design models. This research was supported by the Swiss National Science Foundation through the National Centre of Competence in Digital Fabrication and through Grant No. 200021 200644, the European Research Council under the European Union's Horizon 2020 research and innovation program (Grant No. 866480), and Microsoft Swiss Joint Research Center. Yijiang Huang was supported by an ETH Postdoctoral Fellowship.

REFERENCES

Wolfgang Achtziger and Mathias Stolpe. 2009. Global optimization of truss topology with discrete bar areas—Part II: Implementation and numerical results. *Computational Optimization and Applications* 44, 2 (2009), 315–341.

Grégoire Allaire, Charles Dapogny, Alexis Faure, and Georgios Michailidis. 2017. Shape optimization of a layer by layer mechanical constraint for additive manufacturing. *Comptes Rendus Mathematique* 355, 6 (2017), 699–717.

A Apolinarska, Ralph Bärtschi, Reto Furrer, Fabio Gramazio, and Matthias Kohler. 2016. Mastering the sequential roof. *Advances in architectural geometry* (2016), 240–258.

Aleksandra Anna Apolinarska. 2018. *Complex timber structures from simple elements: Computational design of novel bar structures for robotic fabrication and assembly.* Ph.D. Dissertation. ETH Zurich.

Rahul Arora, Alec Jacobson, Timothy R. Langlois, Yijiang Huang, Caitlin Mueller, Wojciech Matusik, Ariel Shamir, Karan Singh, and David I.W. Levin. 2019. Volumetric Michell Trusses for Parametric Design & Fabrication. In *Proceedings of the 3rd ACM Symposium on Computation Fabrication* (Pittsburgh, PA, USA) *(SCF '19)*. ACM, New York, NY, USA, 13 pages.

Sara Behdad, Leif P Berg, Deborah Thurston, and Judy Vance. 2014. Leveraging virtual reality experiences with mixed-integer nonlinear programming visualization of disassembly sequence planning under uncertainty. *Journal of Mechanical Design* 136, 4 (2014), 041005.

Martin Philip Bendsoe and Ole Sigmund. 2003. *Topology optimization: theory, methods, and applications*. Springer Science & Business Media.

Jan Brütting, Gennaro Senatore, and Corentin Fivet. 2021. Design and fabrication of a reusable kit of parts for diverse structures. *Automation in Construction* 125 (2021), 103614.

Edvard PG Bruun, Sigrid Adriaenssens, and Stefana Parascho. 2022. Structural rigidity theory applied to the scaffold-free (dis) assembly of space frames using cooperative robotics. *Automation in Construction* 141 (2022), 104405.

Michael Bruyneel and Pierre Duysinx. 2005. Note on topology optimization of continuum structures including self-weight. *Structural and Multidisciplinary Optimization* 29, 4 (2005), 245–256.

Richard H Byrd, Jorge Nocedal, and Richard A Waltz. 2006. Knitro: An integrated package for nonlinear optimization. In *Large-scale nonlinear optimization*. Springer, 35–59.

Mario Deuss, Daniele Panozzo, Emily Whiting, Yang Liu, Philippe Block, Olga Sorkine-Hornung, and Mark Pauly. 2014. Assembling Self-Supporting Structures. *ACM Trans. on Graph. (SIGGRAPH Asia)* 33, 6 (2014), 214:1–214:10.

Edsger W Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* 1, 1 (1959), 269–271.

William Doggett. 2002. Robotic assembly of truss structures for space systems and future research plans. In *Proceedings, IEEE Aerospace Conference*, Vol. 7. IEEE, 7–7.

Philipp Eversmann, Fabio Gramazio, and Matthias Kohler. 2017. Robotic prefabrication of timber structures: towards automated large-scale spatial assembly. *Construction Robotics* 1, 1 (2017), 49–60.

Richard L Fox and Lucien A Schmit Jr. 1966. Advances in the integrated approach to structural synthesis. *Journal of Spacecraft and Rockets* 3, 6 (1966), 858–866.

Augusto Gandia, Stefana Parascho, Romana Rust, Gonzalo Casas, Fabio Gramazio, and Matthias Kohler. 2018. Towards automatic path planning for robotically assembled spatial structures. In *Robotic fabrication in architecture, art and design*. Springer, 59–73.

Omprakash K Gupta and Arunachalam Ravindran. 1985. Branch and bound experiments in convex nonlinear integer programming. *Management science* 31, 12 (1985), 1533–1546.

Valentin N. Hartmann, Ozgur S. Oguz, Danny Driess, Marc Toussaint, and Achim Menges. 2020. Robust Task and Motion Planning for Long-Horizon Architectural Construction Planning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 6886–6893. https://doi.org/10.1109/IROS45743.2020.9341502

Valentin N. Hartmann, Andreas Orthey, Danny Driess, Ozgur S. Oguz, and Marc Toussaint. 2023. Long-Horizon Multi-Robot Rearrangement Planning for Construction Assembly. *IEEE Transactions on Robotics* 39, 1 (2023), 239–252. https://doi.org/10.1109/TRO.2022.3198020

Kazuki Hayashi, Makoto Ohsaki, and Masaya Kotera. 2022. Assembly Sequence Optimization of Spatial Trusses Using Graph Embedding and Reinforcement Learning. *Journal of the International Association for Shell and Spatial Structures* 63, 4 (2022), 232–240.

Yijiang Huang, Caelan R Garrett, Ian Ting, Stefana Parascho, and Caitlin T Mueller. 2021. Robotic additive construction of bar structures: Unified sequence and motion planning. *Construction Robotics* 5, 2 (2021), 115–130.

Yijiang Huang, Juyong Zhang, Xin Hu, Guoxian Song, Zhongyuan Liu, Lei Yu, and Ligang Liu. 2016. Framefab: Robotic fabrication of frame shapes. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–11.

Alec Jacobson. 2019. RodSteward: A Design-to-Assembly System for Fabrication using 3D-Printed Joints and Precision-Cut Rods. *Computer Graphics Forum* 38, 7 (2019), 765–774. https://doi.org/10.1111/cgf.13878 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13878

Gwyllim Jahn, Cameron Newnham, Nick van den Berg, Melissa Iraheta, and Jackson Wells. 2020. Holographic construction. In *Impact: Design With All Senses: Proceedings of the Design Modelling Symposium, Berlin 2019*. Springer, 314–324.

Benjamin Jenett, Amira Abdel-Rahman, Kenneth Cheung, and Neil Gershenfeld. 2019. Material–robot system for assembly of discrete cellular structures. *IEEE Robotics and Automation Letters* 4, 4 (2019), 4019–4026.

Caigui Jiang, Chengcheng Tang, Hans-Peter Seidel, and Peter Wonka. 2017. Design and volume optimization of space structures. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–14.

P. Jiménez. 2013. Survey on Assembly Sequencing: A Combinatorial and Geometrical Perspective. *Journal of Intelligent Manufacturing* 24, 2 (2013), 235–250.

Michal Kocvara. 2010. Truss topology design with integer variables made easy. *Optimization Online, Tech. Rep* (2010).

Nitish Kumar, Norman Hack, Kathrin Doerfler, Alexander Nikolas Walzer, Gonzalo Javier Rey, Fabio Gramazio, Matthias Daniel Kohler, and Jonas Buchli. 2017. Design, development and experimental assessment of a robotic end-effector for non-standard concrete applications. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1707–1713.

Weixian Lan and Julian Panetta. 2022. Efficient Layer-by-Layer Simulation for Topology Optimization. In *Proceedings of the 7th Annual ACM Symposium on Computational Fabrication*. 1–11.

Matthijs Langelaar. 2017. An additive manufacturing filter for topology optimization of print-ready designs. *Structural and multidisciplinary optimization* 55 (2017), 871–883.

Samuel Leder, Ramon Weber, Dylan Wood, Oliver Bucklin, and Achim Menges. 2019. Design and prototyping of a single axis, building material integrated, distributed robotic assembly system. In *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS* W)*. IEEE, 211–212.

Juney Lee. 2018. *Computational Design Framework for 3D Graphic Statics*. PhD Thesis. ETH Zurich, Department of Architecture, Zurich. https://doi.org/10.3929/ethz-b-000331210

Michael McEvoy, Erik Komendera, and Nikolaus Correll. 2014. Assembly path planning for stable robotic construction. In *2014 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE, 1–6.

William McGuire, Richard H Gallagher, and H Saunders. 2000. *Matrix Structural Analysis, 2nd Edition*. Vol. 7. Faculty Books.

Daniela Mitterberger, Kathrin Dörfler, Timothy Sandy, Foteini Salveridou, Marco Hutter, Fabio Gramazio, and Matthias Kohler. 2020. Augmented bricklaying. *Construction Robotics* 4, 3 (2020), 151–161.

Roberto Naboni, Anja Kunic, Aljaz Kramberger, and Christian Schlette. 2021. Design, simulation and robotic assembly of reversible timber structures. *Construction Robotics* 5, 1 (2021), 13–22.

William Neveu, Ivan Puhachov, Bernhard Thomaszewski, and Mikhail Bessmeltsev. 2022. Stability-Aware Simplification of Curve Networks. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings*. 1–9.

Stefana Parascho. 2019. *Cooperative robotic assembly: computational design and robotic fabrication of spatial metal structures*. Ph.D. Dissertation. ETH Zürich.

Nico Pietroni, Marco Tarini, Amir Vaxman, Daniele Panozzo, and Paolo Cignoni. 2017. Position-based tensegrity design. *ACM Transactions on Graphics* 36, 6 (2017), 14 pages.

Gramazio Kohler Research. 2018. Roboarch. https://gramaziokohler.arch.ethz.ch/web/e/lehre/360.html.

Peng Song, Chi-Wing Fu, Prashant Goswami, Jianmin Zheng, Niloy J. Mitra, and Daniel Cohen-Or. 2013. Reciprocal Frame Structures Made Easy. *ACM Transactions on Graphics (SIGGRAPH 2013)* 32, 4 (July 2013), 10 pages. Article No. 94.

Mathias Stolpe. 2016. Truss optimization with discrete design variables: a critical review. *Structural and Multidisciplinary Optimization* 53, 2 (2016), 349–374.

Andreas Thoma, Arash Adel, Matthias Helmreich, Thomas Wehrle, Fabio Gramazio, and Matthias Kohler. 2018. Robotic fabrication of bespoke timber frame modules. In *Robotic fabrication in architecture, art and design*. Springer, 447–458.

Yunsheng Tian, Jie Xu, Yichen Li, Jieliang Luo, Shinjiro Sueda, Hui Li, Karl D.D. Willis, and Wojciech Matusik. 2022. Assemble Them All: Physics-Based Planning for Generalizable Assembly by Disassembly. *ACM Trans. Graph.* 41, 6, Article 278 (2022), 15 pages.

Enmei Wang, Shunan Wu, and Zhigang Wu. 2022. Dynamic multi-constrained assembly sequence planning of large space structures considering structural vibration. *Acta Astronautica* 195 (2022), 27–40.

Weiming Wang, Dirk Munro, Charlie CL Wang, Fred van Keulen, and Jun Wu. 2020. Space-time topology optimization for additive manufacturing. *Structural and Multidisciplinary Optimization* 61, 1 (2020), 1–18.

Ziqi Wang, Peng Song, Florin Isvoranu, and Mark Pauly. 2019. Design and Structural Optimization of Topological Interlocking Assemblies. *ACM Transactions on Graphics (SIGGRAPH Asia 2019)* 38, 6 (2019), 1–13. Article No. 193.

Ziqi Wang, Peng Song, and Mark Pauly. 2021. State of the Art on Computational Design of Assemblies with Rigid Parts. *Computer Graphics Forum (Eurographics 2021* 40, 2 (2021), 633–657.

Emily Whiting, Hijung Shin, Robert Wang, John Ochsendorf, and Frédo Durand. 2012. Structural optimization of 3D masonry buildings. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–11.

Rundong Wu, Huaishu Peng, François Guimbretière, and Steve Marschner. 2016. Printing arbitrary meshes with a 5DOF wireframe printer. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–9.

Hironori Yoshida, Takeo Igarashi, Yusuke Obuchi, Yosuke Takami, Jun Sato, Mika Araki, Masaaki Miki, Kosuke Nagata, Kazuhide Sakai, and Syunsuke Igarashi. 2015. Architecture-Scale Human-Assisted Additive Manufacturing. *ACM Trans. Graph.* 34, 4, Article 88 (jul 2015), 8 pages. https://doi.org/10.1145/2766951

Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797* (2016).

Simon Zimmermann, Ghazal Hakimifard, Miguel Zamora, Roi Poranne, and Stelian Coros. 2020. A Multi-Level Optimization Framework for Simultaneous Grasping and Motion Planning. *IEEE Robotics and Automation Letters* 5, 2 (2020), 2966–2972. https://doi.org/10.1109/LRA.2020.2974684