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

1 Linear Finite Element Model of Frame Structures

Our finite element analysis (FEA) supports various bars' cross-sections and material properties. We list all variables used in our FEA formulation below. We choose $-y$ to be the gravity direction.

- L, the bar's length $[m]$
- A, the cross-section area $[m^2]$
- J_{xx}, the polar second moment of inertia in x-axis $[m^4]$
- J_{yy}, the polar second moment of inertia in y-axis $[m^4]$
- J_{zz}, the polar second moment of inertia in z-axis $[m^4]$
- E, the Young's modulus $[kN/m^2]$
- G, the shear modulus $[kN/m^2]$
- D, the density $[kg/m^3]$
- g, the gravitational acceleration $[kN/kg]$
- $\boldsymbol{M}$, the 3×3 transformation matrix from the local to the global coordinate system.
- $\boldsymbol{K}_b^{\text{local}}$, the 12×12 stiffness matrix in the local coordinate system
- $\boldsymbol{K}_b$, the 12×12 stiffness matrix in the global coordinate system
- $\boldsymbol{w}_b$, the 12×1 self-weight vector in the global coordinate system

Let's assume a local coordinate system whose origin is the bar's start point $\boldsymbol{b}^0$ and x-axis is the bar's axial direction $\boldsymbol{b}^1 - \boldsymbol{b}^0$. The local stiffness matrix $\boldsymbol{K}_b^{\text{local}}$ in this coordinate system can be formulated by assembling sub-matrices for four modes of reaction, including axial force, torsional force around the x-axis, as well as bending around y and z-axes. These submatrices are written below as $\boldsymbol{K}^{\text{local}}_{\text{row indices} \times \text{column indices}}$. These are standard formulas derived from quasi-static elastic beam theory and can be found in all structural analysis books, e.g. [McGuire et al.(1982)].

$$x\text{-axial} \quad \boldsymbol{K}^{\text{local}}_{\{0,6\}\times\{0,6\}} = \begin{bmatrix} \frac{EA}{L} & -\frac{EA}{L} \\ -\frac{EA}{L} & \frac{EA}{L} \end{bmatrix}$$

$$x\text{-torisonal} \quad \boldsymbol{K}^{\text{local}}_{\{3,9\}\times\{3,9\}} = \begin{bmatrix} \frac{GJ_{xx}}{L} & -\frac{GJ_{xx}}{L} \\ -\frac{GJ_{xx}}{L} & \frac{GJ_{xx}}{L} \end{bmatrix}$$

$$y\text{-bending} \quad \boldsymbol{K}^{\text{local}}_{\{2,4,8,10\}\times\{2,4,8,10\}} = \frac{EJ_{yy}}{L} \begin{bmatrix} \frac{12}{L^2} & -\frac{6}{L} & -\frac{12}{L^2} & -\frac{6}{L} \\ -\frac{6}{L} & 4 & \frac{6}{L} & 2 \\ -\frac{12}{L^2} & \frac{6}{L} & \frac{12}{L^2} & \frac{6}{L} \\ -\frac{6}{L} & 2 & \frac{6}{L} & 4 \end{bmatrix}$$

$$z\text{-bending} \quad \boldsymbol{K}^{\text{local}}_{\{1,5,7,11\}\times\{1,5,7,11\}} = \frac{EJ_{zz}}{L} \begin{bmatrix} \frac{12}{L^2} & \frac{6}{L} & -\frac{12}{L^2} & \frac{6}{L} \\ \frac{6}{L} & 4 & -\frac{6}{L} & 2 \\ -\frac{12}{L^2} & -\frac{6}{L} & \frac{12}{L^2} & -\frac{6}{L} \\ \frac{6}{L} & 2 & -\frac{6}{L} & 4 \end{bmatrix}$$

The stiffness matrix $\boldsymbol{K}_b$ of bar b in the global coordinate system is:

$$\boldsymbol{K}_b = \begin{bmatrix} \boldsymbol{M} & & & \\ & \boldsymbol{M} & & \\ & & \boldsymbol{M} & \\ & & & \boldsymbol{M} \end{bmatrix}^T \boldsymbol{K}_b^{\text{local}} \begin{bmatrix} \boldsymbol{M} & & & \\ & \boldsymbol{M} & & \\ & & \boldsymbol{M} & \\ & & & \boldsymbol{M} \end{bmatrix}$$

The gravitational load due to the self-weight of bar b in the global coordinate system is represented by lumping the uniformly distributed load to the bar's two ends as $\boldsymbol{w}_b$:

$$\boldsymbol{w}_{\{6,7,8\}} = \boldsymbol{w}_{\{0,1,2\}} = \begin{bmatrix} 0 \\ -0.5ALDg \\ 0 \end{bmatrix}$$

$$\boldsymbol{w}_{\{3,4,5\}} = \frac{L^2}{12} \boldsymbol{M}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \boldsymbol{M} \begin{bmatrix} 0 \\ -0.5ADg \\ 0 \end{bmatrix}$$

$$\boldsymbol{w}_{\{9,10,11\}} = -\frac{L^2}{12} \boldsymbol{M}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \boldsymbol{M} \begin{bmatrix} 0 \\ -0.5ADg \\ 0 \end{bmatrix}$$

2 Branch-and-Bound Solver

In this section, we first give a detailed explanation of our branch-and-bound solver (i.e., Knitro). Following that, we discuss the influence of our solver's parameters on the effectiveness and efficiency of solving our sequence planning formulation. Lastly, we compare the results generated by our HOLISTIC and z-LANDMARK approach from our benchmark.

2.1 Algorithm Explanation

To illustrate how our branch-and-bound method functions, let's consider a simple example of a linear mixed-integer optimization problem with three binary variables.

$$\begin{aligned} \min_{x_0, x_1, x_2} \quad & 3x_0 - 2x_1 + 1.5x_2 \\ \text{s.t.} \quad & x_0 + 0.5x_1 + x_2 \geq 1, \\ & x_1 + x_2 \leq 2, \\ & x_0, x_1, x_2 \in \{0, 1\} \end{aligned} \tag{1}$$

Figure 1 demonstrates the search trees of a brute force method for solving this mixed integer optimization. The search starts with computing the value of the highlighted node in Figure 1(a) by solving a relaxation problem of Equation 1, where all binary constraints are discarded:

$$\begin{aligned} \min_{x_0, x_1, x_2} \quad & 3x_0 - 2x_1 + 1.5x_2 \\ \text{s.t.} \quad & x_0 + 0.5x_1 + x_2 \geq 1, \\ & x_1 + x_2 \leq 2, \\ & 0 \leq x_0, x_1, x_2 \leq 1 \end{aligned} \tag{2}$$

which provides the current best lower bound of the original mixed-integer optimization (Equation 1). Then, the search split the problem into two sub-problems by setting $x_0 = 0$ or 1; see Figure 1(b)(c). The value of the highlighted node in Figure 1(c), for instance, is computed by solving the relaxation optimization in Equation 2 with an additional constraint $x_0 = 1$. Note that the current best lower bound always equals to the minimum value of all nodes in the search frontier. The search continues splitting the sub-problems by setting $x_i = 0$ or 1 until all binary variables are determined. This breadth-first-search inevitably needs to examine all possible combinations of variables except for nodes with infinity value (i.e., infeasible solutions); see Figure 1(e), making it to be less efficient.

More advanced solvers (e.g., Knitro) often use predefined heuristics to compute locally optimal solutions, which helps trim the search tree and reduce computational time. Figure 2 demonstrates the search trees of a more efficient approach for solving the mixed integer optimization in Equation 1. The key strategy to accelerating the search is first to find a feasible binary solution $x_0 = 1, x_1 = 1, x_2 = 0$ using the depth-first-search; see Figure 2(d). Thus, the objective of the best incumbent solution is 1. Then, all children of highlighted nodes in Figure 2(f)(g) are trimmed because their parents already have a large lower bound value than the incumbent solution, saving a significant amount of computation.

2.2 Solver's Parameters

In Knitro 13.1, the search heuristics are governed by the MIP_HEURISTIC_STRATEGY parameter, offering four options (NONE, BASIC, ADVANCED, and EXTENSIVE). Through experimentation, we observed that the NONE heuristic fails to converge in the majority of test cases. The distinctions among the other three heuristics are relatively minor. We favor the ADVANCED heuristic due to its consistent performance. A convergence plot utilizing various MIP heuristics is illustrated in Figure 3.

In addition to search heuristics, the order in which variables are binarized can also have a significant impact on computation time. In Figure 1 and 2, we pre-determine the search order such that x_0 is binarized first, followed by x_1, and then x_2. The search order can be altered by adjusting PRIORITY values in Knitro 13.1. Figure 3(b) displays a comparison of solving the same assembly sequencing problem with and without using PRIORITY values. In this work, we cannot identify a set of priority values that could be effectively applied across all sequencing problems. As a result, we refrained from predefining any PRIORITY values when benchmarking our algorithm. Nonetheless, determining the appropriate priority values is a promising topic for future research.

2.3 Comparisons between HOLISTIC and z-LANDMARK approach

As shown in Figures 7 and 14 in the main content, our z-LANDMARK solver can only find sub-optimal solutions. However, only some models in our benchmark can cause our z-LANDMARK solver to perform sub-optimally. A current limitation of our HOLISTIC solver is that the solver sometimes cannot converge within the specified time limit. This convergence issue becomes more pronounced when h, the number of bars installed simultaneously, is low (e.g., $h < 4$) or when the total number of bars n is high (e.g., $n \geq 40$).

In some scenarios, such as when $h = 6, 10$ and $n < 40$, both our HOLISTIC solver and z-LANDMARK solver can perform effectively. However, due to the convergence issue faced by our HOLISTIC solver in some test models, its average assembly cost is marginally higher than that of our z-LANDMARK solver. For instance, when $h = 6$ and $n < 40$, our HOLISTIC solver results in lower assembly costs for 14 out of 21 models than our z-LANDMARK solver. Nonetheless, its average is still 1% higher than that of the z-LANDMARK solver due to the presence of 3 non-converged sequences. Please refer to Figure 4 for detailed comparisons.

3 Convexity Proof

Our branch and bound solver relies on accurately calculating the lower bound of each tree node, which requires the relaxation problem to be convex. Here, we prove that the relaxation of our temporal coherent sequential topology optimization is a convex optimization problem. Because all constraints are linear, we only need to prove its objective is a convex function.

Theorem 3.1 *The compliance $C(\boldsymbol{\rho})$ is a convex function of the bar thickness $\boldsymbol{\rho}$.*

We can prove this theorem by computing the gradient and Hessian of C with respect to its variables $\boldsymbol{\rho}$.

$$\frac{\partial C}{\partial \boldsymbol{\rho}} = \left(\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\rho}}\right)^T \boldsymbol{U} - \frac{1}{2}\boldsymbol{U}^T \frac{\partial \boldsymbol{K}}{\partial \boldsymbol{\rho}} \boldsymbol{U} \tag{3}$$

$$\frac{\partial^2 C}{\partial \boldsymbol{\rho}^2} = \left(\frac{\partial \boldsymbol{U}}{\partial \boldsymbol{\rho}}\right)^T \boldsymbol{K} \left(\frac{\partial \boldsymbol{U}}{\partial \boldsymbol{\rho}}\right) \tag{4}$$

The Hessian of C is clearly a semi-positive matrix and therefore C is a convex function with respect to its variables. Because our objective function is a summation of all compliances, the relaxation problem of our sequential topology optimization is convex.

4 Greedy & Beam Search

This paragraph describes our implementation of greedy for determining the sequence in both single or multiple bar assembly processes. Since greedy search can be viewed as a specific case of beam search, we focus on explaining the beam search method instead.

We use our state graph to guide the search. Our method stores all candidate graph nodes in a priority queue (e.g., Fibonacci heap) and sorts them in the ascending order of their assembly costs. Starting by pushing the start node into the queue with 0 cost, our method only expands the top X nodes of the priority queue. The remaining nodes are discarded.

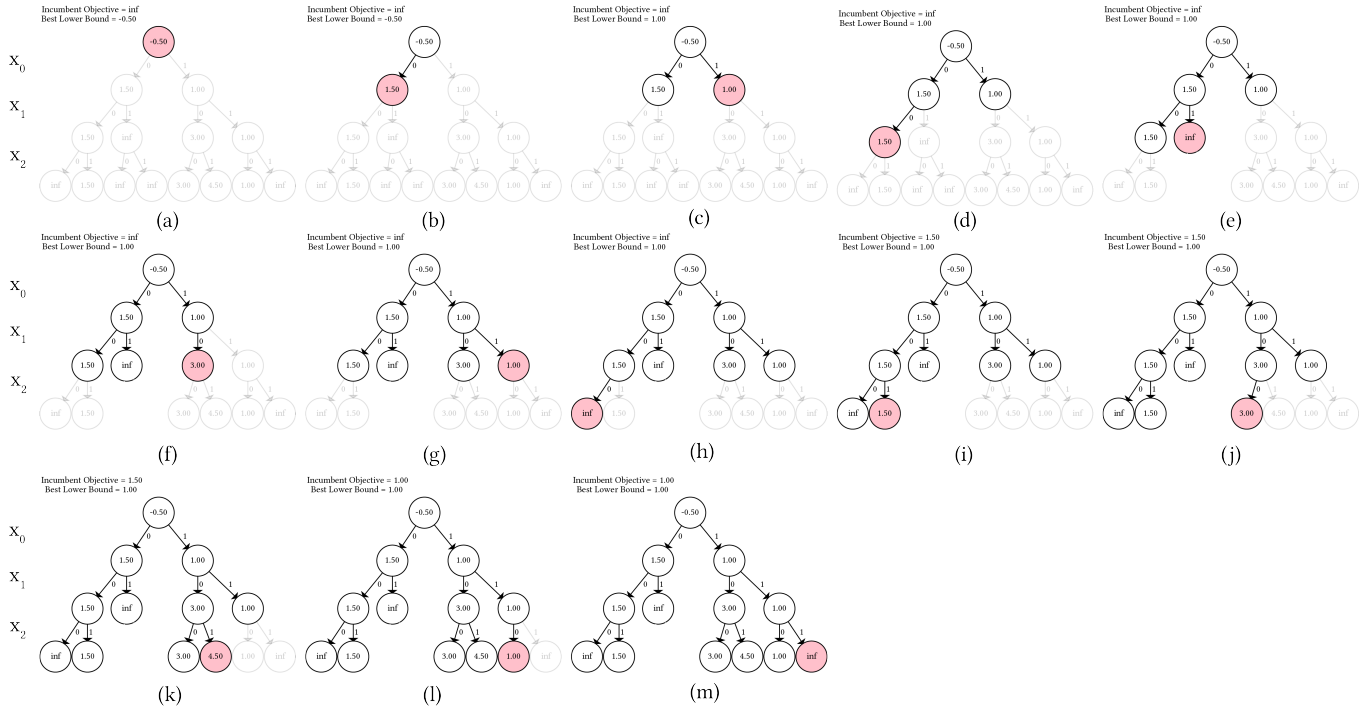

Figure 1: *An illustration of solving the mixed integer optimization specified in Equation 1 through a breadth-first binary tree search. Unvisited nodes are shown in transparent colors. (a) The initial step of the binary tree search involves solving the relaxed problem (Equation 2). (b) The node value is determined by solving the relaxed optimization with an extra constraint,* $x_0 = 0$*. (c) The current best lower bound is* 1.0 *as* $1.0 = \min(1.5, 1.0)$*. The top node's value of* -0.5 *is disregarded since it is no longer part of the search frontier. (e) The descendants of the highlighted node are pruned as the current additional binary constraints (*$x_0 = 0, x_1 = 1$*) already lead to an infeasible optimization. (i) A feasible solution is founded (*$x_0 = 0, x_1 = 0, x_2 = 1$*). However, the current best lower bound still remains less than the incumbent objective, causing the solution to be non-globally optimal (*$1.0 < 1.5$*).(l) A globally optimal solution is founded (*$x_0 = 1, x_1 = 1, x_2 = 0$*) since the current best lower bound is equivalent to the incumbent objective.*

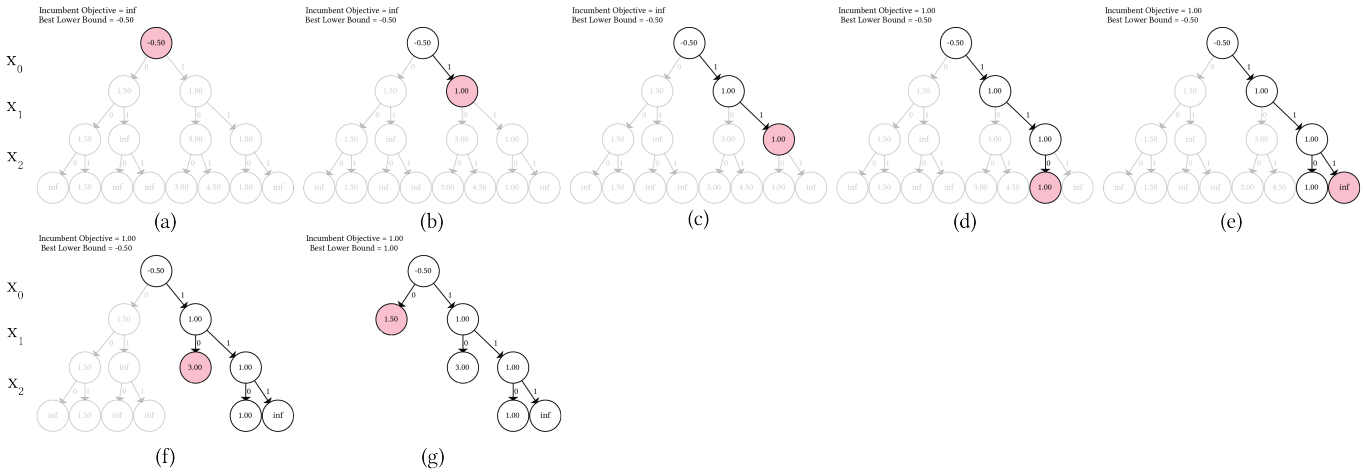

Figure 2: *A visualization of solving the mixed integer optimization defined in Equation 1 using a tree search with heuristics. (d) Utilizing a depth-first-search, the search initially identifies a feasible solution (*$x_0 = 1, x_1 = 1, x_2 = 0$*). (f) The descendants of the highlighted node are disregarded, and the search stops going deeper, as finding a better solution is impossible given that the lower bound of the highlighted node is already greater than the current best incumbent solution, with* $3.0 > 1.0$*. (g) The global optimal solution is confirmed, as the current best lower bound is equal to the incumbent objective.*

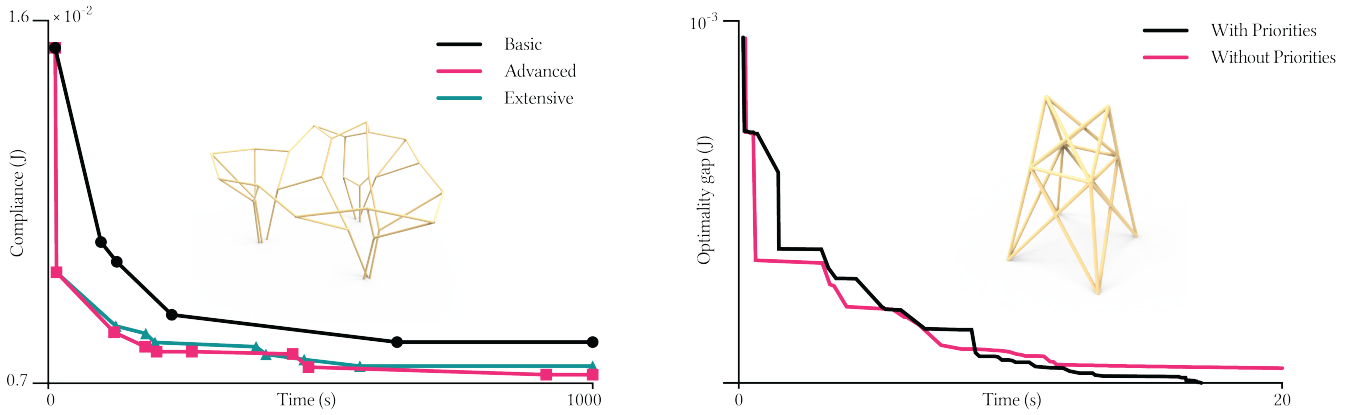

Figure 3: *(a) A comparison in optimization convergence using three distinct heuristic settings in Knitro 13.1 for determining the assembly sequence of the* ROBOARCH *model, with the installation of* 4 *parts at a time. The* ADVANCED *heuristic demonstrates the best performance. (b) A comparison in optimization convergence for determining the assembly sequence of the* TOWER *model, with the installation of* 5 *parts at a time. Knitro enables the assignment of priority values P_i^t to each variable ρ_i^t for our temporal coherent topology optimization, facilitating a reduction in computational time. Higher priority values result in branching the respective variables first. In this illustration, we assign $P_i^t = -t$, compelling the solver to prioritize sub-assemblies with fewer bars. Employing this priority strategy, the optimization converges in* 17 *s, compared to* 735 *s without utilizing the strategy. However, this specific priority strategy does not consistently generate good results across all test examples.*

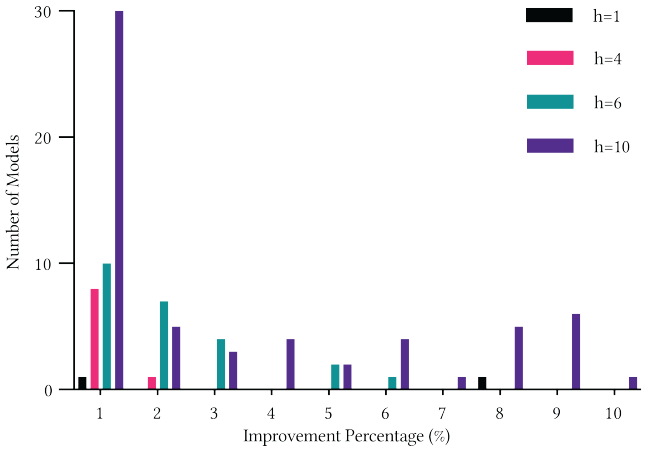

Figure 4: *The histogram shows the relative ratio of the assembly cost found by the* HOLISTIC *and the z-*LANDMARK *approach from Table 1 and 2 in the main content. We have excluded test models for which our* HOLISTIC *solver clearly failed to converge. The models presented in this figure represent* $9.5\%, 42.9\%, 32.4\%$ *and* 82.4% *of the test models for* $h = 1, 4, 6,$ *and* 10*, respectively.*

Algorithm 1 Sequence Planning with Beam Search

```
v_0.cost = 0
v_i.cost = ∞, ∀i > 0
priority queue Q ← v_0
while Q ≠ ∅ do
    new priority queue Q' ← ∅
    for k ← 1 to X do
        u ← Q.pop()
        if u = v_final then
            return u
        for v ∈ V and (u, v) ∈ E do
            c ← u.cost + v.compliance
            if v.cost > c then
                v.prev ← u          ▷ store assembly sequence
                v.cost ← c          ▷ update assembly cost
                Q' ← v
    Q = Q'
```

We cache the cost of the nodes that have already been explored to save time from performing duplicated FEA for computing costs. The newly explored nodes are then added to the queue. We repeat this procedure until the final node is reached. We present a pseudocode of the baseline algorithm in the Algorithm 1. We refer to this beam search solver with a beam width X as the X-BEAM method. This beam search implementation degenerates to the greedy search when setting $X = 1$.

Table 1 shows that the performance of our beam search implementation is heavily affected by the beam width X. A large beam width X helps reduce the assembly cost but can significantly increase the program's running time and memory consumption. Practically, our beam search implementation with $X = 100$ works well in generating assembly plans for frame structures with $h = 1$. However, our beam search implementation runs significantly slower when $h \geq 3$ and $n \geq 50$. This is because the number of successors of almost every search node grows exponentially with h, which significantly slows down the search process. Since beam search is unable to manage most multi-bar assembly processes effectively, we only include their benchmarking results in this

supplementary material.

Interestingly, integrating this beam search method with our z-LANDMARK technique can reduce overall computation time for multi-bar assembly processes. As shown in Table 1, our z-LANDMARK method initially computes two temporally coherent landmarks, splitting the entire planning problem into three sub-problems. We subsequently address each of these sub-problems using our beam search method. Due to the significant reduction in undecided bars within each sub-problem, our beam search can effectively determine the assembly sequence for each sub-problem. However, for single bar assembly processes, although our 2-LANDMARK solver can significantly shorten the time required to solve each sub-problem between landmarks, it still has a relatively large overhead when computing the landmarks. As a result, the average computation time of our 2-LANDMARK solver is longer compared to the 100-BEAM method in the single-bar settings.

5 Frame structure dataset

To benchmark our algorithms, we manually selected 100 models from the Thingi10k dataset [Zhou and Jacobson(2016)] and then used libigl's [Jacobson et al.(2018)] edge-collapse algorithm to reduce their total number of edges. Figure 5 depicts these models, demonstrating a large variety of geometrical and topological features. The dataset, along with all the other models and results shown in the paper, is available at `https://github.com/KIKI007/sequencer_benchmark`

References

Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. https://libigl.github.io/.

William McGuire, Richard H Gallagher, and H Saunders. 1982. Matrix structural analysis. (1982).

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

Table 1: *This table examines how beam width X can affect the quality and computational time required to determine the assembly sequence for both single and multiple-bars-at-a-time assembly processes using our 100-model dataset.* BEAM *refers to our beam search implementation, while* LANDMARK *denotes our landmark approach that calculates two temporally coherent landmarks and addresses each subproblem with our beam search. The first column shows the maximum step size h. The third column shows the beam width X used for computing the assembly sequences. The table entries are marked with "-" if running the corresponding algorithm takes more than* 1000*s.*

h	Methods	X	Assembly Cost			Time (s)		
			Small	Medium	Large	Small	Medium	Large
1	BEAM	1	1	1	1	0.01 (0.00∼0.01)	0.02 (0.01∼0.04)	0.06 (0.02∼0.25)
		10	0.80 (0.48∼1.00)	0.75 (0.35∼0.99)	0.81 (0.44∼1.00)	0.06 (0.03∼0.08)	0.12 (0.08∼0.21)	0.47 (0.18∼1.7)
		100	0.78 (0.41∼0.99)	0.67 (0.34∼0.99)	0.75 (0.35∼1.00)	0.56 (0.26∼0.74)	1.2 (0.73∼2.1)	5.0 (1.7∼19.4)
		1000	0.77 (0.40∼0.99)	0.65 (0.34∼0.99)	0.71 (0.29∼1.00)	5.4 (2.5∼7.2)	11.4 (7.2∼20.4)	55.3 (17.5∼235)
		10000	0.77 (0.40∼0.99)	0.64 (0.34∼0.99)	0.69 (0.28∼1.00)	53.6 (23.0∼71.9)	116 (73.0∼238)	706 (184∼2419)
	LANDMARK	1	0.89 (0.56∼1.11)	0.80 (0.42∼1.11)	0.82 (0.50∼1.07)	1.5 (0.33∼3.1)	4.6 (0.44∼20.0)	21.2 (1.7∼59.5)
		10	0.80 (0.51∼1.04)	0.69 (0.37∼1.07)	0.73 (0.36∼1.03)	1.7 (0.37∼3.3)	4.8 (0.62∼20.2)	21.6 (1.9∼61.0)
		100	0.79 (0.50∼1.04)	0.67 (0.36∼1.07)	0.70 (0.28∼1.03)	1.9 (0.55∼3.6)	5.4 (1.2∼21.3)	25.7 (3.0∼79.4)
		1000	0.79 (0.50∼1.04)	0.67 (0.36∼1.07)	0.69 (0.28∼1.03)	2.6 (0.56∼5.1)	10.8 (3.8∼37.8)	68.8 (13.8∼247)
		10000	0.79 (0.50∼1.04)	0.67 (0.36∼1.07)	0.69 (0.28∼1.03)	2.8 (0.54∼6.3)	34.1 (4.8∼145)	455 (89.2∼1863)
4	BEAM	1	1	1	1	0.71 (0.12∼1.4)	2.3 (0.31∼12.0)	103 (1.8∼1281)
		10	0.88 (0.72∼1.00)	0.85 (0.63∼0.99)	-	7.3 (1.0∼16.1)	24.5 (5.6∼126)	-
		100	0.86 (0.66∼1.00)	-	-	76.6 (11.4∼169)	-	-
	LANDMARK	1	0.95 (0.75∼1.05)	0.90 (0.61∼1.09)	0.90 (0.66∼1.13)	1.5 (0.28∼3.0)	4.7 (0.59∼19.7)	25.4 (2.2∼91.4)
		10	0.90 (0.68∼1.05)	0.85 (0.50∼1.05)	0.83 (0.54∼1.03)	1.7 (0.38∼3.3)	5.3 (1.1∼21.6)	56.4 (5.8∼383)
		100	0.89 (0.68∼1.05)	0.84 (0.46∼1.05)	0.82 (0.54∼1.03)	2.5 (0.54∼4.6)	9.9 (3.2∼34.2)	332 (26.1∼3082)

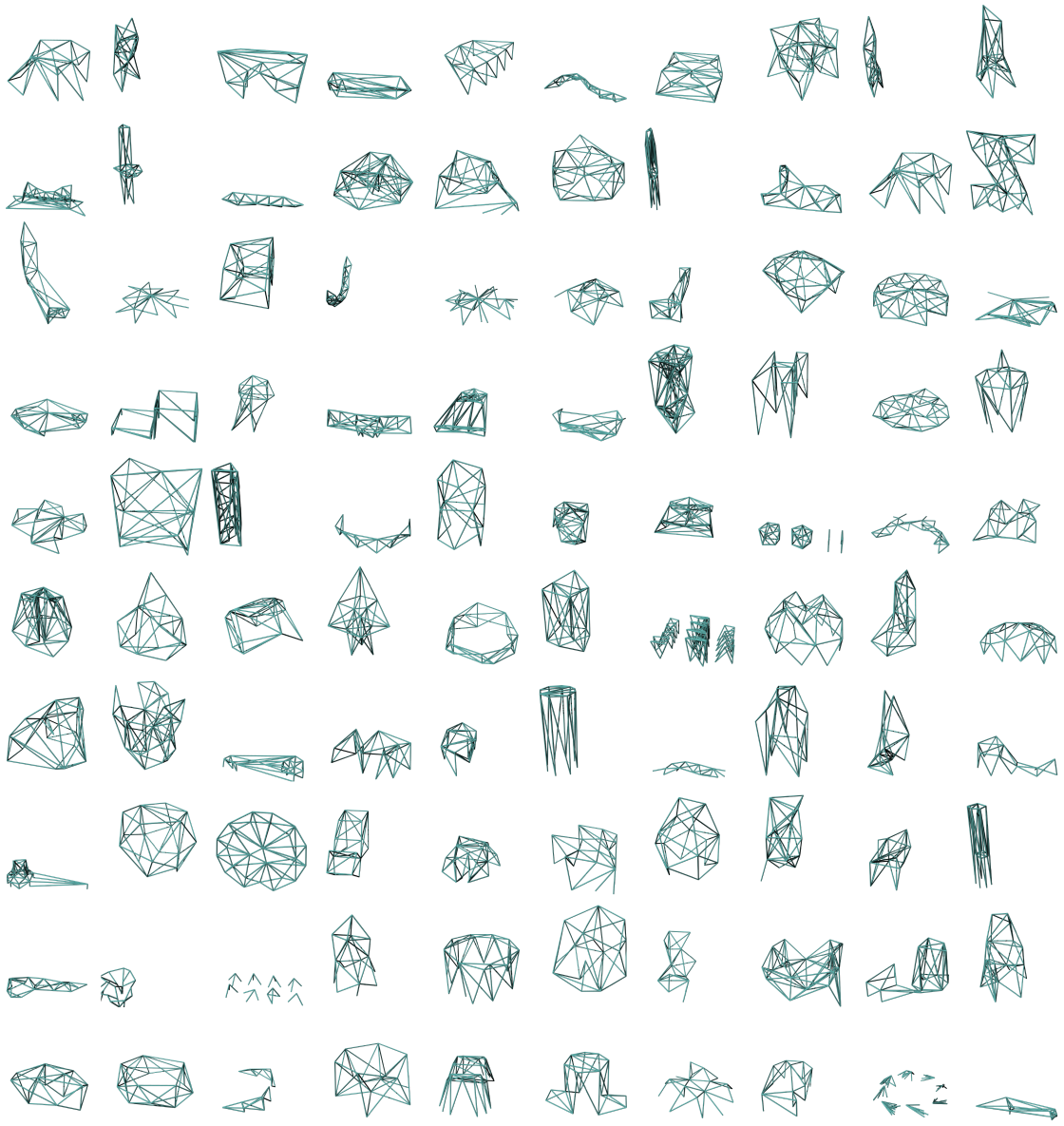

Figure 5: *Our* 100*-model dataset of freeform frame structures. These structures are derived from the Thingi10k dataset (see Section 7 in the main paper).*