Design and Volume Optimization of Space Structures

CAIGUI JIANG, King Abdullah University of Science and Technology (KAUST) and MPI for Informatics
CHENGCHENG TANG, King Abdullah University of Science and Technology (KAUST) and Stanford University
HANS-PETER SEIDEL, MPI for Informatics
PETER WONKA, King Abdullah University of Science and Technology (KAUST)

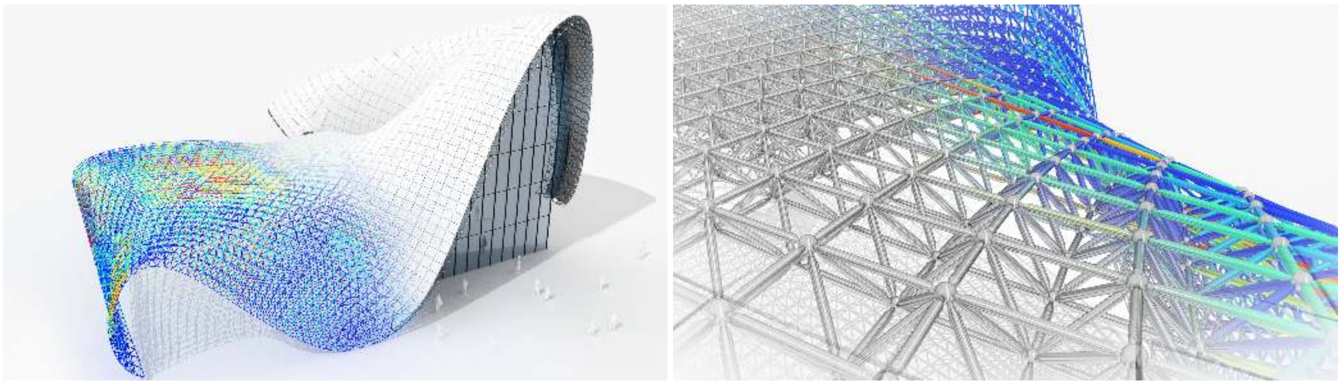


Fig. 1. Left: A statically sound space structure designed and optimized with our framework, motivated by the real architectural project shown in Figure 2. Right: The space structure is constructed with six types of customized beams to minimize the total volume of the material used for beams while maintaining moderate manufacturing complexity. Here, a hotter color indicates a larger beam cross-section area. Our framework automatically determines the optimal cross-section areas of the six types of beams as well as the assignment of beam types.

We study the design and optimization of statically sound and materially efficient space structures constructed by connected beams. We propose a systematic computational framework for the design of space structures that incorporates static soundness, approximation of reference surfaces, boundary alignment, and geometric regularity. To tackle this challenging problem, we first jointly optimize node positions and connectivity through a nonlinear continuous optimization algorithm. Next, with fixed nodes and connectivity, we formulate the assignment of beam cross sections as a mixed-integer programming problem with a bilinear objective function and quadratic constraints. We solve this problem with a novel and practical alternating direction method based on linear programming relaxation. The capability and efficiency of the algorithms and the computational framework are validated by a variety of examples and comparisons.

CCS Concepts: • **Computing methodologies → Shape modeling**;

General Terms: Design, Algorithms, Optimization

Additional Key Words and Phrases: Architectural geometry, static equilibrium, structural optimization, optimization

ACM Reference format:
Caigui Jiang, Chengcheng Tang, Hans-Peter Seidel, and Peter Wonka. 2017. Design and Volume Optimization of Space Structures. *ACM Trans. Graph.* 36, 4, Article 1 (July 2017), 14 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073619

1 INTRODUCTION

Space structures, also called space frames or space frame structures, are elegant and materially efficient *truss*-like structures consisting of beams (*two-force members*) connected at nodes. Space structures are desirable and often necessary in industrial design and architectural construction. The design and optimization of space structures present many challenges, especially for designs with complex geometries [Freund 2004].

In industrial design, space structures have been widely used for structures that should be both lightweight and statically sound, such as bikes, cars, or airplanes. A major advantage of space structures is their static soundness with limited material usage. In many situations, space structures also enable simple manufacturing processes by assembling or welding beams or bars. Moreover, the design space for space structures is rich, allowing for the possibility of the emergence of elegant structures that support designs with highly customized shapes. Similar ideas have also been extended to 3D printing and personalized design and fabrication.

In architectural construction, space structures often serve as statically sound supportive structures that approximate intended shapes or that underlie desired freeform surfaces. In many prominent projects, they are not visible. However, they are essential for the realization of the architectural design. For example, the Heydar Aliyev Cultural Center in Azerbaijan designed by Zaha Hadid has an underlying space structure as shown in Figure 2.

Space structures can also remain visible. Aesthetic considerations, such as simplicity and regularity, may influence the choice to make space structure visible. Visible space structures are not limited to

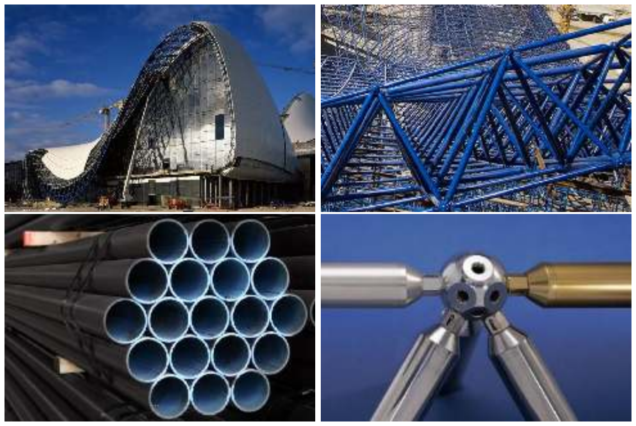


Fig. 2. The Heydar Aliyev Cultural Center (top-left) in Baku, Azerbaijan was designed by Zaha Hadid. The supporting space structure (top-right) underlying the freeform surface was constructed by MERO-TSK using circular hollow section tubular beams (bottom-left) and the KK-Ball Node System (bottom-right).

freeform architectural designs of stadia or cultural centers. They are commonly seen on bridges, towers, and even playground domes.

Despite their universal applicability, most of the current conventional design processes for space structures are based on manually exploring different variations using interactive editing and customized scripting. There are usually many iterations of design and verification to optimize the connectivities, node positions, and cross sections of the beams. To simplify and automate this design process, we collaborated with structural engineers who work on architectural projects in industry. We determined that structural engineers must achieve the following four goals in designing space structures. (Goal 1) First, the structure should be statically sound. This means that the structure is in force equilibrium with axial forces along the beams without bending moments. (Goal 2) Second, the structure should be constructed with regularly arranged beams and nodes to be aesthetically pleasing. (Goal 3) Third, the structure should approximate a given designed shape, e.g., a freeform architectural surface. (Goal 4) Fourth, the cost of the structure should be minimized. The most important factor associated with cost is material usage. A cost-effective space structure consists of beams with variable cross sections, given that most material is used for the beams. Because the beams are usually manufactured by extrusion (aluminium) or bending and welding (steel), followed by cutting, many beams should share the same cross section.

Here, we propose a systematic framework to design and optimize statically sound space structures that approximate freeform surfaces. To achieve the goals stated above, our framework allows a user to explore different configurations, optimize the node positions and connectivity, and adjust the beam cross sections. Our contributions to the design and optimization of space structures include:

- A novel system for the design and optimization of space structures that achieves the goals required by structural engineers.
- A break down of the overall problem into subproblems that are formulated into manageable optimization problems.
- A nonlinear optimization algorithm that jointly optimizes node positions and connectivity.
- A practical optimization algorithm that efficiently tackles a challenging mixed-integer programming problem with a bilinear objective function and quadratic constraints.

2 PREVIOUS WORK

Computer Graphics. To bridge the gap between digital content creation and physical realization, graphics researchers have began to incorporate fabrication considerations in computational design processes, leading to fabrication-aware design. Many problems in this field share two distinct yet interrelated themes: the assurance of static soundness and the reduction of manufacturing cost. Static soundness has been explored in the context of 3D printing [Langlois et al. 2016; Prévost et al. 2013; Wang et al. 2013], furniture [Umetani et al. 2012], gridshell structures [Pietroni et al. 2015], and self-supporting surfaces [Block and Ochsendorf 2007; de Goes et al. 2013; Liu et al. 2013; Panozzo et al. 2013; Tang et al. 2014; Vouga et al. 2012]. To reduce manufacturing cost, researchers looked at the use of off-the-shelf parts [Schulz et al. 2014], simplification of geometric components [Liu et al. 2006; Tang et al. 2016], and construction using repetitive elements [Fu et al. 2010; Huard et al. 2014; Jiang et al. 2014]. In line with these two themes, we study the design of statically sound space structures of minimal volume to reduce production cost.

Industrial Practice. In industrial practice, the design and optimization of truss structures mainly follow a forward process in which manually created structures are evaluated computationally for static and dynamic soundness. Numerous software packages are available for structural analysis, including SAP2000, Robot Structural Analysis, SOFiSTiK, Etabs, and Dlubal. Despite their prominence and popularity, these computational packages do not guide the improvement of the design. Since the involved optimization problems are tough to solve, currently only simple and general methods, such as evolutionary optimization are commercially available. For example, combining Karamba 3D, Grasshopper, and Rhinoceros creates a simple method. Clearly, a lot more research is needed to develop commercially viable tools.

Structural Engineering. Designing trusses that support imposed external loads with minimal material usage is a fundamental topic in structural engineering. It was first examined by Michell [1904] for the hypothetical case of smooth, curvilinear, and infinitesimally thin beams. This theoretical study was derived from the principle of virtual work, following the argument of James Clerk Maxwell [1870] on force diagrams. For computation and analysis, the most influential method in the research community is the ground structure method (GSM) [Dorn 1964; Zegard and Paulino 2014, 2015], which provides an approximation of optimal Michell trusses by using a finite number of beams. GSM starts from an over-complete structure and removes excessive beams to minimize the total volume. It has two similar formulations: elastic and plastic. The correspondence between these two formulations is summarized in the additional materials Our method is built upon the plastic formulation.

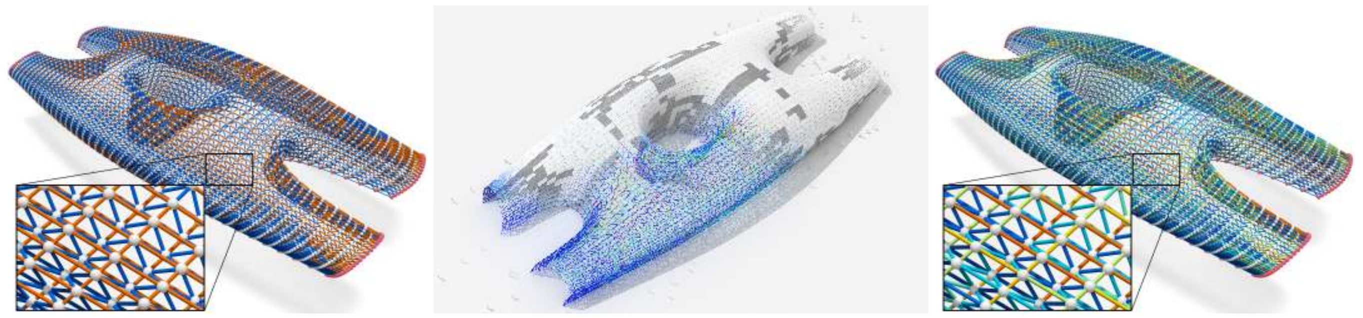


Fig. 3. Two different construction solutions of a statically sound space structure approximating the train station model. The solution on the left uses two types of customized tubular beams, and the solution in the middle and the right uses six types. Similar to Figure 1, the beams are color coded by their cross-section types, where hotter colors represent stronger beams. For aesthetics, these beams have the same outer radius, as the cross-section areas are controllable by the thicknesses of the circular hollow sections (shown in Figure 2, bottom-left).

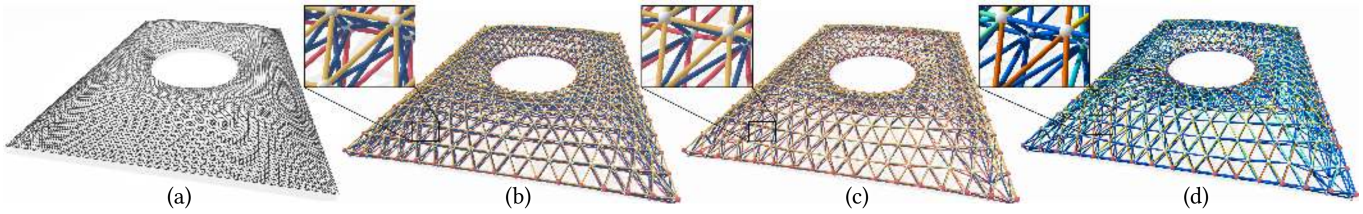


Fig. 4. Framework overview: Based on an input reference mesh (a), our system provides a set of tools for creating an initial structure (b). After joint optimization of node positions and connectivity (c), we further adjust the beam types to minimize the total volume of material used under the condition that the structure should be constructed with beams of a limited number of cross section types that could be customized (d).

Despite its elegance and influence, traditional GSM has three major limitations. First, GSM fixes the node positions in advance and thus limits the solution space. We relax this restriction and allow adjustable node positions in § 4.2. Smith et al. [2002] also noticed this shortcoming when studying truss structures in virtual reality, albeit without considering the other two limitations. Second, GSM formulation yields results with self-intersecting beams. To overcome this limitation, we supply an energy term that discourages incompatible beam configurations in § 4.2 and illustrate the effect of such an energy term through comparisons. Finally, GSM assumes that the cross sections of beams can be independently and arbitrarily adjusted, which is impractical for batch production for which the cost is often unaffordable without utilizing repetitive elements.

Therefore, multiple researchers [Achtziger and Stolpe 2007; Kanno and Guo 2010; Rasmussen and Stolpe 2008; Stolpe and Svanberg 2003] restrict beam cross sections to a preassigned set and tackle the problem using mixed-integer programming, targeting a globally optimal solution. Alternatively, many meta-heuristic approaches have been proposed for the same problem, such as genetic algorithms [Kawamura et al. 2002], ant colony optimization [Kaveh et al. 2008], particle swarm optimization [Li et al. 2009], and teaching-learning-based optimization [Camp and Farshchin 2014]. However, both the mixed integer programming methods and meta-heuristics methods have been demonstrated only on very small models. We choose to compare to [Rasmussen and Stolpe 2008] as a representative algorithm, because it computes a globally optimal solution (through a parallel cut-and-branch method). As shown in the comparisons in § 5, determining the beam cross sections in advance is still inefficient and there is potential for huge volume savings realized by our method. Further, the fast computation speed of our method enables us to create examples with an order of magnitude more beams than used in previous work.

3 OVERVIEW

We provide users a framework to create space structures, optimize for static soundness, and minimize the total volume of material used. As shown in Figure 4, the framework comprises three stages: 1) connectivity enumeration, ranking, and editing, 2) joint optimization of node positions and connectivity, and 3) discrete optimization of beam cross sections.

Connectivity Enumeration, Ranking, and Editing. The input to our system includes a reference model of a finely tessellated triangle mesh with desired boundary curves on which the boundary vertices are allowed to glide. We provide a set of tools to generate the initial connectivity for the space structure:

- Coarse mesh enumeration, ranking, and editing.
- Subdivision for mesh refinement and pattern generation.
- Construction of multi-layer structures.
- Addition of alternative beams to the structure.

To provide enough degrees of freedom for optimization, we allow connectivities with invalid configurations such as edge intersections at this stage. These issues are resolved in the following stages.

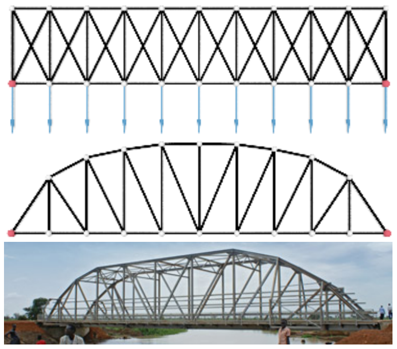


Fig. 5. Top: an initial over-complete 2D bridge configuration with uniform loads. Middle: a 2D bridge design after joint optimization of node positions and connectivity. Bottom: a similar truss bridge constructed in the real world.

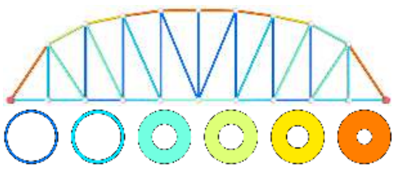


Fig. 6. Discrete optimization assigns customized beam cross section types to minimize material usage while ensuring static equilibrium.

Optimization of Node Positions and Connectivity. Based on the initial structure with possibly conflicting beam connections, we jointly optimize the node positions and connectivity following three steps. As the first step, axial force densities of beams are estimated based on proper load assumptions. Next, we apply a nonlinear continuous optimization to adjust the node positions and axial force densities for material efficiency, geometric proximity, structural regularity, static soundness, and connectivity. Finally, excessive beams identified by nearly zero axial force densities are removed by thresholding, which leaves a structure with valid connectivity. An illustrative example is presented in Figure 5.

Discrete Optimization of Beam Cross Sections. While constructing space structures, we minimize the total material consumption by adjusting the area of the cross sections of the beams. At the same time, space structures should be constructed with limited types of beams with adjustable cross sections of each type, as demonstrated in Figure 6. This problem involves discrete variables (assignments of types), and continuous variables (cross-section areas of beams). With an optimization scheme alternating between continuous and discrete variables, we can successfully tackle this problem, overcoming the restrictions of general-purpose, mixed-integer programming solvers.

4 OPTIMIZATION FRAMEWORK

The inputs to our framework include a reference surface given as a fine triangle mesh, M, and boundary curves represented as polylines or splines, $C^i, i = 1 \dots n^C$. Our goal is to design and optimize a space structure, S, that consists of a set of nodes (vertices), V, connected by a set of beams (edges), E. The supported nodes are $V^f \subset V$, and the boundary vertices are $V^B \subset V$, which may also be supported. We denote $\overline{V} \subset V$ as the nodes of the outer layer that should approximate the reference surface. In single-layer structures, $\overline{V} = V$.

4.1 Connectivity Enumeration, Ranking, and Editing

As connectivity is an essential element in space-structure design, we provide a set of connectivity modeling tools, as demonstrated in Figure 7. This set of tools includes connectivity enumeration, interactive editing, connectivity refinement, and beam addition. The connectivity is further optimized in § 4.2.

Coarse-mesh Enumeration. Coarse-mesh enumeration can be used for the outer layer. For space structures with quadrilateral base meshes, the connectivities of the base layer meshes are generated based on [Peng et al. 2014]. Triangle meshes are created by CVT-based remeshing [Yan et al. 2009].

Interactive Editing. Interactive editing allows the user to sketch a desired coarse mesh. A user can add, remove, and relocate vertices and edges. A user also can edit a mesh by adding or removing polylines and relocating singular vertices.

Connectivity Refinement. A user can select from a set of subdivision and procedural rules to refine the mesh. Simple subdivision rules include Loop and Catmull-Clark. Procedural refinement rules include the construction of hexagonal meshes and semi-regular patterns [Jiang et al. 2015]. In multi-layer structures, derivation rules are applied based on single-layer meshes [Chen and Lui 2005] as shown in Figure 8.

Beam Addition. To enable connectivity optimization in the next stage, additional beams are added to the space structure while keeping the number and locations of nodes fixed. We provide multiple ways to add beams, e.g., two intersecting diagonals for each quadrilateral generated based on offsetting are added, as shown in Figure 9.

Figure 7 illustrates different connectivities created using a combination of the methods described above on the British Museum model. All of the space structures are optimized using the later stages of our framework and are structurally sound.

4.2 Optimization of Node Positions and Connectivity

After an initial configuration of the space structure is generated, node positions and connectivity are jointly optimized for material efficiency, geometric proximity, static soundness, and connectivity validity, as illustrated in Figure 10. The goal of joint optimization is to find proper node positions and valid connectivity, which are required inputs for the discrete optimization of cross sections of beams in § 4.3. Optimization of node positions and connectivity follows three steps. Firstly, axial forces are estimated under given structural assumptions, e.g., loads proportional to the corresponding areas of the shell. Next, an energy term encoding the total volume, geometric properties, static equilibrium, and combinatorial restrictions of edges is minimized through a continuous nonlinear optimization algorithm. Finally, based on the results of the continuous optimization, thresholding is applied to remove redundant beams while ensuring the validity of the connectivity and the static soundness of the structure.

4.2.1 Force Initialization. A space structure must be statically sound. In the following, we first describe our structural assumptions and then introduce how axial forces are initialized.

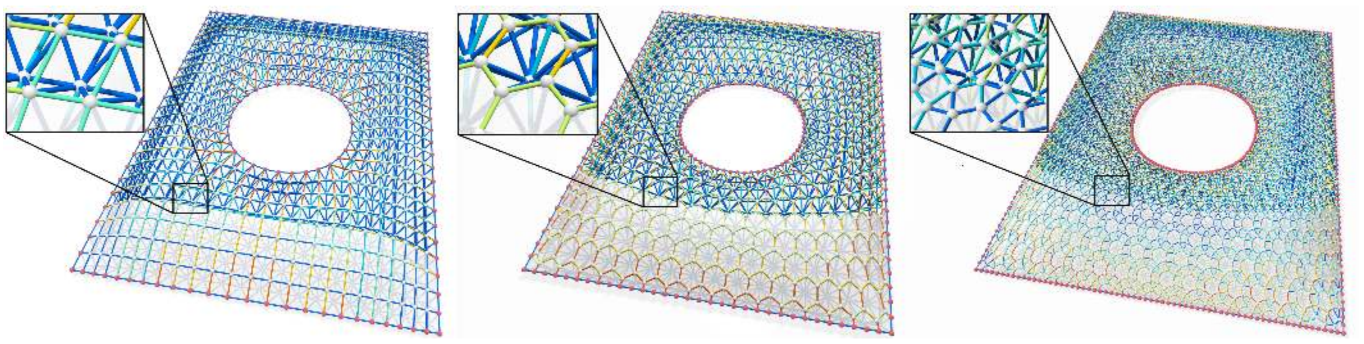


Fig. 7. To enrich the design space, our framework provides a user with a set of tools to create base meshes with a variety of connectivities. In addition to the structure based on a triangle mesh in Figure 4, we show space structures approximating the same British Museum model with a quadrilateral base mesh (left), a hexagonal base mesh (middle), and a semi-regular pattern consisting of triangles, quadrilaterals, and hexagons (right). Each of the structures is constructed with tubular beams with six types of customized cross sections.

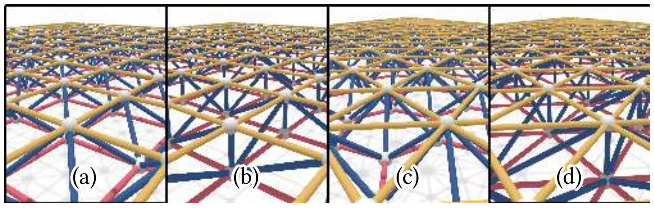


Fig. 8. Derivation of double-layer space structures based on offsetting a quad mesh with (a) and without (b) dualization and a triangle mesh with (c) or without (d) dualization.

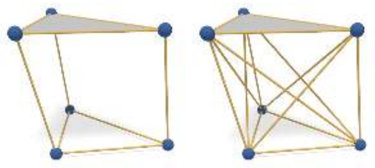


Fig. 9. Based on a construction from derivation rules (left), additional beams are added (right) for connectivity optimization.

Structural Assumptions. Our structural assumptions closely follow previous work in computer graphics and structural engineering. In space structure design, we aim for structures in static equilibrium such that the axial forces are in balance with the loads. We assume that the nodes are sufficiently strong for structures in static equilibrium, especially as the torques born by them are eliminated by balanced axial forces. Similar to previous work, e.g., [Vouga et al. 2012], the dead loads of the nodes on the outer layer are assumed to be proportional to the influence areas, i.e., the areas of the dual cells, of the shell supported by the space structure, as shown in Figure 11. Here, we also assume that the weights of the beams are lower than the weights of the panels that the structure has to support. Alternative assumptions do not essentially change the workflow.

Approximation of Axial Forces. To proceed to the next stages of computation, all the variables should be properly initialized. The missing variables are the axial force densities. The force densities, w_{ij}, are the axial forces per unit length defined on each beam. The axial forces at each node, $\mathbf{v}_i$, should balance the load, $\mathbf{l}_i$, imposed on the node. These forces are precomputed according to the structural assumptions described above. This is equivalent to the minimization of the energy term encoding the force equilibrium:

$$\sum_{j:\ \{i,j\}\in E} w_{ij}(\mathbf{v}_j - \mathbf{v}_i) = -\mathbf{l}_i, i = 1, \ldots, |V|. \tag{1}$$

If there is no solution, the system is solved in a least-squares sense. If there are multiple solutions, a least-norm solution is computed. In the next stage of computation, the node locations are treated as variables optimized together with the axial forces. Note that working with axial force densities instead of axial forces simplifies highly nonlinear force balance conditions into bilinear equations when node positions are variables [Vouga et al. 2012].

4.2.2 *Continuous Optimization.*

After all the variables are properly initialized, the node coordinates should be further adjusted with the axial forces. We model an objective function consisting of six terms to be minimized: closeness to the reference surface (E_{close}), boundary alignment ($E_{boundary}$), static equilibrium (E_{static}), total volume (E_{volume}), geometric regularity ($E_{reg.}$), and combinatorial validity ($E_{comb.}$).

Closeness to Reference Surfaces. The outer layer of the space structure is required to approximate the reference shape provided as the fine triangle mesh, M. The closeness of a vertex, $\mathbf{v}_i \in \overline{V}$, to the reference surface, M, is constraining $\mathbf{v}_i$ to move only on the tangent plane associated with its closest point, $\mathbf{v}_i^*$, on the reference mesh, M:

$$E_{close} = \sum_{\mathbf{v}_i\in\overline{V}} \left((\mathbf{v}_i - \mathbf{v}_i^*)\cdot \mathbf{n}_i^*\right)^2 . \tag{2}$$

Here, the vector, $\mathbf{n}_i^*$, is the unit normal vector of the tangent plane. Figure 12 illustrates the effects of the closeness term.

Boundary Alignment. We also require that the boundary vertices stay on the desired boundary curves, C^i, by enforcing that a boundary vertex, $\mathbf{v}_i \in V^B$, can only move along the tangent line at its projected point, $\mathbf{v}_i^\dagger$, on the reference curve:

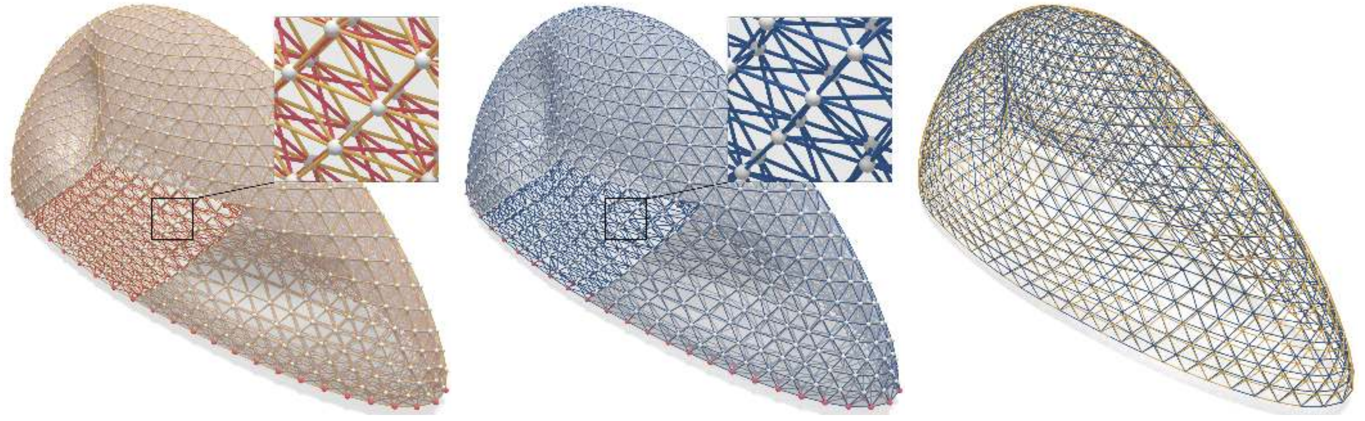


Fig. 10. Based on an initial configuration of a space structure with infeasible intersections (left), joint optimization of node positions and connectivity creates a structure (middle) with optimized node positions and valid connectivity. A comparison of the upper layer before and after optimization shows that the vertices are slightly shifted from their initial positions (right).

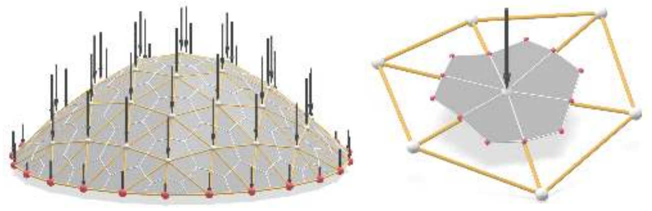


Fig. 11. A space structure (left) supporting a shell like surface is in force balance under appropriate structural assumptions, e.g., the load at every node is proportional to the area of its dual cell (right). Red dots in the left indicate supported nodes.

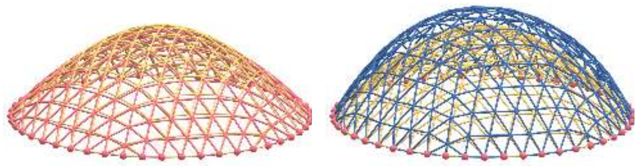


Fig. 12. Optimization of a single-layer space structure with the energy term enforcing closeness (left) provides a better approximation to the reference surface colored in yellow. In contrast, optimization without the closeness term (right) drives the structure away from the initial design to further reduce the total volume of material used.

$$E_{boundary} = \sum_{\substack{\mathbf{v}_i \in V^B \\ j \in \{1,2\}}} \left((\mathbf{v}_i - \mathbf{v}_i^\dagger) \cdot \mathbf{n}_{i,j}^\dagger \right)^2 . \tag{3}$$

Here, $\mathbf{n}_{i,1}^\dagger$ and $\mathbf{n}_{i,2}^\dagger$ are unit vectors that are orthogonal to each other and to the tangent vector at the projected point, $\mathbf{v}_i^\dagger$.

Static Equilibrium. The axial forces are related to both the total volume of the material consumption and the force balance. Absolute values of the force density terms, $|w_{ij}|$, are needed to represent the total volume. We therefore introduce slack variables, u_{ij}^+ and u_{ij}^-, whose squares represent the magnitudes of compressive and tensile forces respectively: $(u_{ij}^+)^2 = \max(0, w_{ij})$, $(u_{ij}^-)^2 = \max(0, -w_{ij})$. Here, squared slack variables ensure the nonnegativity of the quantities without requiring additional inequality constraints [Tang et al. 2014]. With these new variables, $w_{ij} = (u_{ij}^+)^2 - (u_{ij}^-)^2$ and $|w_{ij}| = (u_{ij}^+)^2 + (u_{ij}^-)^2$. The energy term for static equilibrium, E_{static}, is:

$$\sum_{i \in V} \left(\sum_{j:\ \{i,j\} \in E} \left((u_{ij}^+)^2 - (u_{ij}^-)^2 \right) (\mathbf{v}_j - \mathbf{v}_i) + \mathbf{l}_i \right)^2 . \tag{4}$$

Total Volume. The minimal cross-section areas of the beams are linearly related to the absolute values of axial forces. Each solution of axial forces corresponds to a different assignment of a cross-section area of a beam. With the introduced slack variables, the total volume of material can be written as:

$$E_{volume} = \sum_{i,j:\ \{i,j\} \in E} \left((u_{ij}^+)^2 + (u_{ij}^-)^2 \right) \|\mathbf{v}_j - \mathbf{v}_i\|^2 . \tag{5}$$

Geometric Regularizers. For aesthetics, a space structure should also be regular. For a space structure with a triangle mesh as its base mesh, a uniformly weighted Laplacian term is enforced on each layer for smoothness. For a structure with a quadrilateral mesh as its base mesh, polyline fairness energy is applied. For a space structure constructed based on a semi-regular pattern, symmetry-based regularizers proposed by Jiang et al. [2015] are used.

Combinatorial Validity. Based on an initial structure with dense connections and infeasible intersections, beams should be removed for structural validity, as illustrated in Figures 9 and 14. Here, we describe how to incorporate combinatorial constraints to remove invalid beam intersections, e.g., two diagonals of a quadrilateral cannot be selected at the same time. In principle, combinatorial choices of beams are expressible as Boolean variables, $p_i \in \{0, 1\}$, indicating the selection of the i-th beam. Disallowing two intersecting edges to be selected at the same time corresponds to a logic statement,

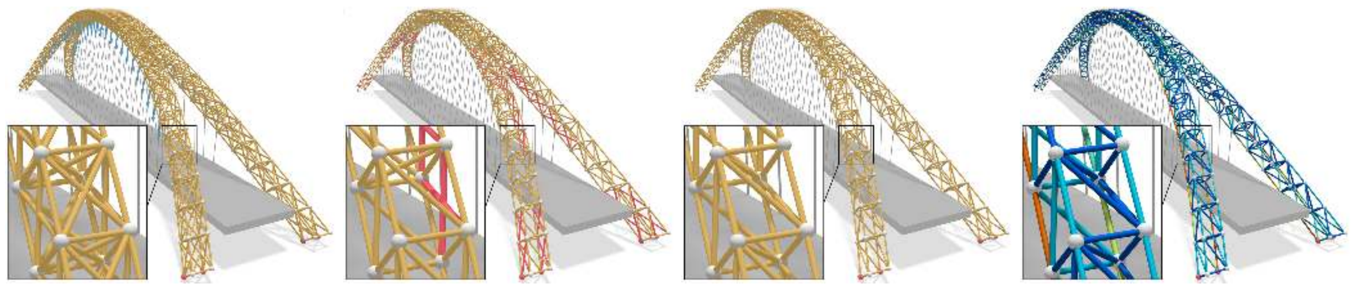


Fig. 13. For a densely connected bridge with infeasible intersections (left), the ground structure method fails to remove all the invalid intersections (second from left). In comparison, optimization with the combinatorial validity term, $E_{comb.}$, ensures that the final structure has a valid connectivity (second from right), which could be further optimized discretely for minimized volume in the next stage (right).

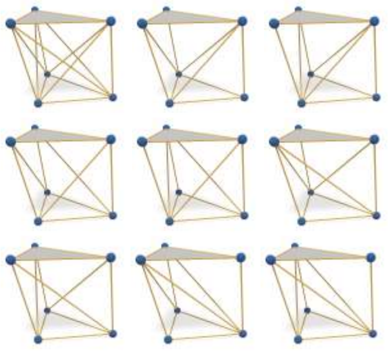


Fig. 14. Based on a densely connected space structure with conflicting beam intersections (top-left), the combinatorial validity energy, $E_{comb.}$, resolves the conflicts and guides the computation to an optimized structure among the valid configurations (others).

$\neg(p_i \wedge p_j)$. As p_i and p_j are Boolean variables, a constraint, $p_i + p_j \leq 1$ or $p_i \cdot p_j = 0$, would be sufficient to guarantee that the two edges cannot be selected simultaneously.

To simplify the computation, instead of working directly with Boolean variables, we apply *continuous relaxation* and use force densities, w_i. If the i-th and the j-th edges are exclusive, the product of their force densities, $w_i \cdot w_j$, should be zero. Enforcing exclusivenesses among edges is therefore equivalent to minimizing an energy term summing absolute values of the corresponding products:

$$E_{comb.} = \sum_{i,j:\ \{i,j\} \in Ex} |w_i \cdot w_j|. \tag{6}$$

Here, Ex indicates the set of index pairs for edges that are exclusive. With auxiliary variables u_i^+ and u_i^- introduced previously, $|w_i|$ is replaced by $(u_i^+)^2 + (u_i^-)^2$:

$$E_{comb.} = \sum_{i,j:\ \{i,j\} \in Ex} \left((u_i^+)^2 + (u_i^-)^2\right) \cdot \left((u_j^+)^2 + (u_j^-)^2\right). \tag{7}$$

Minimizing the energy term, $E_{comb.}$, enforces the assumption that at least one of the two beams related by each bilinear term should have zero force density. As a result, optimizing the total energy with $E_{comb.}$ ensures that the combinatorial constraints are respected. This energy term also distinguishes our computational approach from the traditional ground structure method [Dorn 1964; Zegard and Paulino 2014], which often generates results with invalid connectivity configurations, as illustrated in Figure 15.

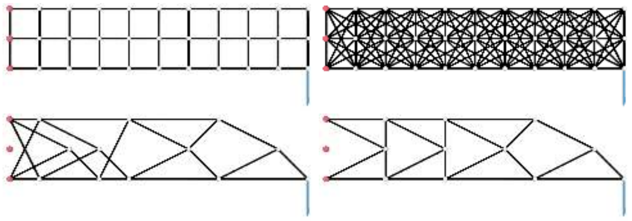


Fig. 15. Based on a densely connected structure (top-right) generated from an initial configuration (top-left), structural optimization with combinatorial validity energy term (bottom-right) successfully removes invalid beam intersections, in contrast to the traditional ground structure method (bottom-left).

Optimization Algorithm. The sum of the energy terms is solved by a quasi-Newton method [Liu and Nocedal 1989], based on an implementation of the Hybrid Limited-memory Broyden-Fletcher-Goldfarb-Shanno (HLBFGS) method developed by Liu et al. [2009]. For the sum of energy terms, $E_{sum} = \sum_i \lambda_i E_i$, we choose higher weights, 1, for closeness, boundary alignment, combinatorial validity, and force equilibrium. We choose lower weights, 0.01, for the other terms.

4.2.3 Removing Excessive Beams. Finally, with the computed force densities as input, the edges with axial forces close to zero are removed based on thresholding under the condition of connectivity validity. In our computation, we use a threshold value of $10^{-6} \cdot \bar{a}$, where $\bar{a}$ is the allowed maximal axial force. Axial forces are recomputed for force equilibrium after edge removal to ensure that the structure after beam removal is still statically sound. In Figure 13, we show a 3D example for the effect of topology optimization for choosing candidate beams out of an over-complete graph, where two intersecting beams are present in the initial configuration.

At this stage, we have not considered the practical construction requirement that the beams should have repetitive cross-section areas. Therefore, after joint optimization of node positions and connectivity, we take the structures as valid input and assign the cross-section areas of the beams through discrete optimization in § 4.3.

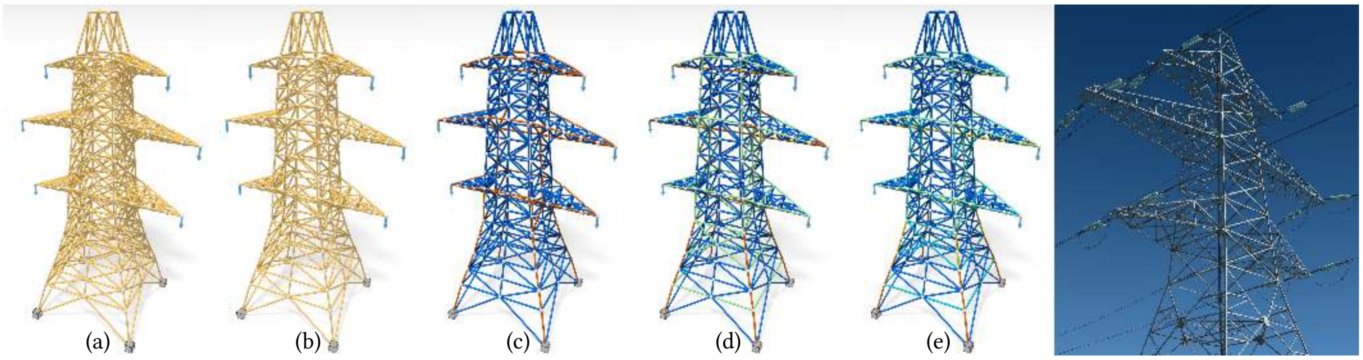


Fig. 16. Electrical transmission tower design: after connectivity initialization in § 4.1 (a) and joint optimization of node positions and connectivity in § 4.2 (b), discrete optimization assigns 2 (c), 3 (d), and 6 (e) cross sections of beams, following procedures in § 4.3.

4.3 Discrete Optimization of Cross Sections of Beams

The solution obtained from the previous stage assumes that the cross-section areas of the beams can be arbitrarily and continuously adjusted. While the total material consumption is reduced, such a solution imposes a high cost for beam customization, as every beam has to be manufactured with a different factory setting. However, a lot of material would be wasted if all beams had the same cross section. Therefore, as a compromise, we create structures based on a small number, e.g., 2, 3, or 6, of cross sections.

To keep the problem manageable, the previous nonlinear formulation needs to be simplified before discrete variables are introduced. To reduce the complexity, we fix the node positions and solely focus on the axial forces and cross-section areas of the beams. Such simplification is feasible as the structures we study are *statically indeterminate* (or *hyperstatic*). This means that the solution space of axial forces in force equilibrium generally has a nontrivial dimension for fixed nodes and given loads as illustrated in Figure 17. In the infinite space of valid solutions, there is typically one that has minimal volume for beams with varying cross sections. However, this solution is not necessarily optimal when beams are chosen from a fixed set of cross sections. Therefore, we cannot simply solve a continuous problem and take the solution as a basis for the discrete problem.

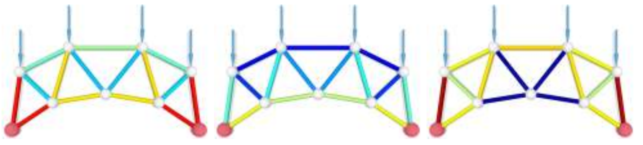


Fig. 17. Different solutions of axial forces in static equilibrium for a statically indeterminate (hyperstatic) structure with given nodes and loads. The beams are color coded according to the magnitudes of axial forces, where red beams bear greater forces.

The natural formulation of the optimization is a mixed integer programming problem. However, there is no generic method to solve such a problem with a modest number of variables and constraints. In the following, we first present the mixed integer programming problem. We then discuss our novel solution to tackle this problem in a practical manner.

Mixed Integer Programming Formulation.

$$\underset{x_{ij},\bar{a}_j,s_i}{\text{minimize}} \quad \sum_i l_i \sum_{j=1}^{k} \bar{a}_j x_{ij} \tag{8}$$

$$\text{subject to} \quad B^T \mathbf{s} = -\mathbf{f} \tag{8a}$$

$$\sum_{j=1}^{k} x_{ij}\bar{a}_j + s_i \geq 0; \qquad i = 1, \ldots, |E| \tag{8b}$$

$$\sum_{j=1}^{k} x_{ij}\bar{a}_j - s_i \geq 0; \qquad i = 1, \ldots, |E| \tag{8c}$$

$$\sum_{j=1}^{k} x_{ij} \leq 1; \qquad i = 1, \ldots, |E| \tag{8d}$$

$$x_{ij} \in \{0, 1\}; \quad j = 1, \ldots, k, \quad i = 1, \ldots, |E|. \tag{8e}$$

Here, the coefficients, l_i, are the lengths of the beams. The scalars, $\bar{a}_j, j = 1, \ldots, k$, are the bounds of the axial forces of the k-beam types. The assignment of beams types is reflected by the integer variable, x_{ij}, which is 1 if the i-th beam is assigned with the j-th type, and 0 otherwise. Each element of $\mathbf{s} \in \mathbb{R}^{|E|}$ is a signed scalar value of the axial force of a beam, proportional to the beam's length and the force density: $s_i = w_i l_i$.

The linear constraints, $B^T \mathbf{s} = -\mathbf{f}$, in Equation 8, for balance of axial forces is equivalent to Equation 1 for force densities. The matrix, $B^T \in \mathbb{R}^{3|V|\times|E|}$, is called the *nodal equilibrium matrix*. It represents the connectivity of the space structure, converting axial forces on beams to the resultant forces. More details about this matrix are given in the additional materials. The vector, $\mathbf{f} \in \mathbb{R}^{3|V|}$, represents all the loads, $\mathbf{l}_i, i = 1, \ldots, |V|$. The inequality constraints, Equations 8b and 8c, bound the axial force, s_i, to the range allowed by the selected cross-section area. Equations 8d and 8e indicate that at most one cross-section area can be assigned to each beam.

Difficulties of Mixed Integer Programming. The seemingly simple formulation presented above turns out to be remarkably difficult to solve with a generic solver for mixed integer programming. This is due to the fact that both the objective function and the constraints are quadratic. Besides, the objective function is generally not positive definite. With discrete variables, the problem becomes even more challenging.

There are three sets of variables: the axial forces of beams, s_i, the cross-section areas of beam types $\overline{a}_j$, and the assignment of types for beams x_{ij}. We propose to break the overall problem into individual subproblems and solve them in an alternating scheme. Studying the structure of the problem, we derive the following three subproblems:

- Sp-1: Fix $\overline{a}_j$, and solve for s_i and x_{ij}.
- Sp-2: Fix s_i, and solve for $\overline{a}_j$ and x_{ij}.
- Sp-3: Fix x_{ij}, and solve for s_i and $\overline{a}_j$.

In the following, we discuss how these three subproblems are formulated and solved. We then discuss the overall algorithm comprising these three subproblems.

Sp-1 (fix $\overline{a}_j$): When the cross-section areas of the k-beam types, $\overline{a}_j$, are given, Equation 8 is reduced to a standard mixed integer *linear* programming problem in terms of **s** and x_{ij}. However, directly applying a generic solver has very poor scalability, and it soon becomes impractical even for a problem of moderate scale, as shown in Section 5.

Linear programming relaxation. To further reduce the complexity, we relax the integer variables, x_{ij}, to real variables [Agmon 1954]. Instead of requiring x_{ij} to be either 0 or 1, we allow them to be continuously adjustable between 0 and 1:

$$\begin{aligned} \underset{x_{ij}, s_i}{\text{minimize}} \quad & \sum_i l_i \sum_{j=1}^{k} \overline{a}_j x_{ij} \\ \text{subject to} \quad & (8a) - (8d) \\ & x_{ij} \geq 0. \end{aligned} \tag{9}$$

In Equation 9, each sum, $\sum_{j=1}^{k} \overline{a}_j x_{ij}$, can assume values between 0 and $\max(\overline{a}_j, j = 1, \ldots, k) := a_{max}$ with non-unique choices of x_{ij}. Therefore, we eliminate x_{ij} by introducing $a_i = \sum_{j=1} \overline{a}_j x_{ij}$ and further simplify the problem to the following continuous linear programming problem:

$$\underset{a_i, s_i}{\text{minimize}} \quad \sum_{i=1}^{|E|} l_i a_i \tag{10}$$

$$\text{subject to} \quad B^T \mathbf{s} = -\mathbf{f} \tag{10a}$$

$$a_i + s_i \geq 0; \quad i = 1, \ldots, |E| \tag{10b}$$

$$a_i - s_i \geq 0; \quad i = 1, \ldots, |E| \tag{10c}$$

$$a_i \leq a_{max}; \quad i = 1, \ldots, |E|. \tag{10d}$$

Sp-2 (fix s_i): When axial forces are given, we seek the cross-section areas of the beam types, $\overline{a}_j$, and beam-type assignments, x_{ij}. We sort the beams according to the absolute values of axial forces, $|s_i|, i \leq |E|$, in a nondecreasing order and relabel them accordingly. Next, we look for optimal cuts at the indices, $m_t \in \mathbb{Z}^+, t = 1 \ldots k-1$, so that the volume is minimized:

$$\underset{m_t \in \mathbb{Z}^+,\ t=1\ldots k-1}{\text{minimize}} \quad \sum_{i=1}^{k} \left(\sum_{j=m_{i-1}+1}^{m_i} l_j |s_{m_i}| \right). \tag{11}$$

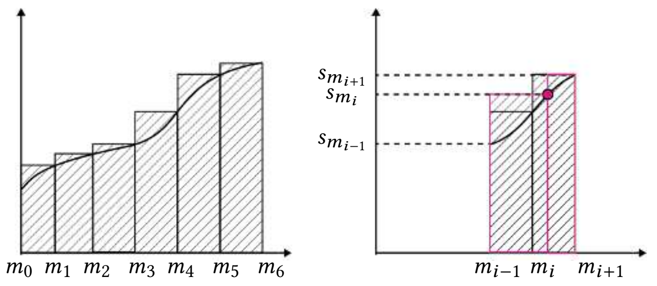

m_0 m_1 m_2 m_3 m_4 m_5 m_6
$s_{m_{i+1}}$
s_{m_i}
$s_{m_{i-1}}$
m_{i-1} m_i m_{i+1}

Fig. 18. Left: In Sp-2, we sort the beams according to the absolute values of axial forces, $|s_i|$, $i \leq |E|$, in a nondecreasing order and seek the optimal cuts to determine the beam types. Right: We start from uniformly sampled cuts and adjust each cut between its neighbors while fixing the other cuts iteratively.

By definition, the boundaries of the cuts are $m_0 = 0$, and $m_k = |E|$. The internal cuts, $m_i, i = 1 \ldots k-1$, are initialized equidistantly between m_0 and m_k, as shown on the left side of Figure 18. Next, each individual cut is adjusted by sweeping in the range bounded by its two neighboring cuts, while fixing the other cuts, as shown on the right side of Figure 18. This routine of seeking the optimal cuts with given balanced forces generally converges in fewer than 10 iterations.

Sp-3 (fix x_{ij}): Next, we consider the case when the assignment of beams is fixed in Equation 8. For better clarity, we denote $\Phi_j := \{i | x_{ij} = 1\}$ as the set of beams that is assigned with the j-th type:

$$\begin{aligned} \underset{s_i, \overline{a}_j}{\text{minimize}} \quad & \sum_{j=1}^{k} \left(\sum_{i \in \Phi_j} l_i \right) \overline{a}_j \\ \text{subject to} \quad & B^T \mathbf{s} = -\mathbf{f} \\ & \overline{a}_1 + s_i \geq 0,\ \overline{a}_1 - s_i \geq 0;\ i \in \Phi_1 \\ & \ldots \\ & \overline{a}_k + s_i \geq 0,\ \overline{a}_k - s_i \geq 0;\ i \in \Phi_K \end{aligned} \tag{12}$$

As coefficients of $\overline{a}_j$, the sums, $\sum_{i \in \Phi_j} l_i$, are constants. This problem is thus a standard linear programming problem.

Overall Algorithm. The three subproblems are assembled together to create an overall practical algorithm constituting three stages. First, as a preprocessing step, we determine the bounds of a_{max}, which could be used as input for Sp-1 (fix $\overline{a}_j$). Next, as the main procedure, the algorithm alternates between Sp-1 (fix $\overline{a}_j$) and Sp-2 (fix s_i) to find the optimal beam type assignment, x_{ij}. Finally, as a post-processing step, we use Sp-3 (fix x_{ij}) to further adjust the cross-section areas of each type, $\overline{a}_j$.

Pre-processing: In Sp-1, after linear programming relaxation, the maximal allowed cross section area, a_{max}, is the only fixed variable. To find the range of feasible inputs for Equation 10 of Sp-1, we look for the lower and upper bounds of a_{max}: $(a_{max})^{min}$ and $(a_{max})^{max}$. The lower bound, $(a_{max})^{min}$, could be found by a linear programming formulation similar to Equation 10:

$$\begin{aligned} \underset{s_i,\ a_{max}}{\text{minimize}} \quad & a_{max} \\ \text{subject to} \quad & (10a) - (10d). \end{aligned} \tag{13}$$

As a special case, when there is only one customizable beam type allowed, the optimization goal is precisely finding the lower bound of a_{max}, as shown in above equation.

To find the upper bound, $(a_{max})^{max}$, we solve Equation 10 without constraint 10d, and we take the maximal value of $a_j, j = 1, \ldots, |E|$. Any choice of a_{max} greater than $(a_{max})^{max}$ does not activate constraint 10d, giving the same solution.

Main procedure: Here, we alternate between Sp-1 and Sp-2. The output of Sp-1 includes axial forces, s_i, which could be directly used as input for Sp-2. However, there is no readily available output from Sp-2 that could be used as input for Sp-1. Therefore, we sample different values of a_{max} for Sp-1, compute corresponding optimal values, and estimate a gradient to update a_{max}. Note that Sp-3 is excluded here as its output lacks meaningful input for Sp-1.

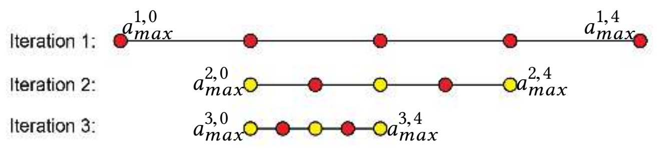


Fig. 19. We apply Sp-1 and Sp-2 to narrow the range for the best choice of a_{max}. Starting from $a^{1,0}_{max} = (a_{max})^{min}$ and $a^{1,4}_{max} = (a_{max})^{max}$ in Iteration-0, we uniformly sample $a^{1,1}_{max}$, $a^{1,2}_{max}$, and $a^{1,3}_{max}$ to compute the total volumes based on both Sp-1 and Sp-2. Then, we update the range according to the minimal value to continue and repeat the procedure. Here, the red dots indicate the values requiring new computation.

Our approach is illustrated in Figure 19: We start the search at $a^{0,0}_{max} = (a_{max})^{min}$ and $a^{1,4}_{max} = (a_{max})^{max}$. At the m-th iteration, we uniformly sample $a^{m,1}_{max}$, $a^{m,2}_{max}$, and $a^{m,3}_{max}$ to divide the range between $a^{m,0}_{max}$ and $a^{m,4}_{max}$. Among the five samples including the boundaries, we find the n-th value, $a^{m,n}_{max}$, assuming the smallest optimal volume from Sp-2, and update the range for the next iteration accordingly: $a^{k+1,0}_{max} = a^{k,max(0,n-1)}_{max}$ and $a^{k+1,4}_{max} = a^{k,min(4,n+1)}_{max}$. We stop this process when the total volume no longer decreases, usually after 5-10 iterations.

Post-processing: Finally, we use the computed type assignment from the previous stage to run Sp-3 to further refine the choice of cross-section areas for the beam types.

5 RESULTS AND DISCUSSION

In this section, we present example designs, discuss quantitative results, and compare our method with an alternative approach.

5.1 Example Designs

We illustrate construction solutions for a double-layer space structure design motivated by the real project shown in Figure 1.We present a design of the train station model in Figure 3, and space structures with base meshes with different connectivities in Figures 4 and 7. In addition, we show a bridge in Figure 13, an electrical transmission tower in Figure 16, a result resembling a constructed stadium in Figure 20, a freeform pillar in Figures 21 and 22, a structure approximating the Lilium Tower model in Figure 24, a pentagon-mesh-based structure of the Soumaya model in Figure 25, and a tunnel based on a semi-regular pattern in Figure 26.

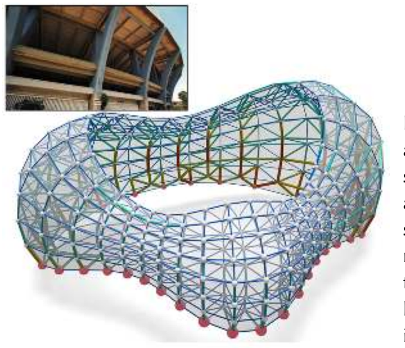


Fig. 20. Our framework automatically specifies the strong supporting pillars of a quad-based double-layer space structure. The assignment coincides with structural designs observed in real life like the stadium shown in the inset photo.

5.2 Quantitative Evaluation

Our framework is implemented in C++ with customized data structures. We use the HLBFGS Library developed by Liu et al. [2009] for the nonlinear optimization procedure and Mosek [ApS 2016] for linear programming. We run our tests on a workstation with an Intel Xeon X5550 2.67GHz processor. The most time-consuming step is the discrete optimization of the beams' cross sections (§ 4.3). In Table 1, we report the computational time of the discrete optimization and achieved total volume for each model when the number of customized beam types is 1, 2, 3, 6, and arbitrary.

5.3 Comparisons

Resolving Self-intersections. As mentioned in § 2, one of the major limitations of GSM is that it cannot resolve self-intersections. We show that the joint optimization of node positions and connectivity (§ 4.2) does resolve conflicting beam configurations, which GSM cannot, in Figures 13 and 15.

k-means clustering. As a baseline method, we consider k-means clustering of axial forces computed by the ground structure method in the continuous setting. However, we observe that our method consistently outperforms k-means clustering by a large margin. For example, for the train station model (Figure 3), after multiple trials with different initializations, the minimal achieved volumes of k-means clustering for 2, 3, and 6 types of beams are (326.70, 240.60, and 180.61), compared with (242.22, 194.69, and 157.23), our results. There are two main disadvantages of k-means clustering. On the one hand, it relies on the optimal assignment of axial forces in the continuous setting, which is often suboptimal when discrete variables are introduced. On the other hand, it lacks the formulation of an objective function for volume minimization.

Comparisons with Preassigned Cross-Section Areas. We chose [Rasmussen and Stolpe 2008] as a representative method of the current state of the art for extending GSM to a fixed number of cross-sections. We compare the performance of our method with [Rasmussen and Stolpe 2008], using the examples, solutions, and running times provided in their paper. In addition, we also compare to our solution obtained by solving the unrestricted continuous GSM problem first and then rounding-up to the minimal supportable beam.

We test these three methods on the benchmark structures with the same geometry and physical parameters used in [Rasmussen and Stolpe 2008]. The optimized structures and total volumes are

Structure	Beams	Arbitrary Beam Types Available		1 Customized Beam Type		2 Customized Beam Types		3 Customized Beam Types		6 Customized Beam Types		Fig.
		Volume	Time(s)	Volume	Time(s)	Volume	Time(s)	Volume	Time(s)	Volume	Time(s)	
Tower	821	20.09	0.18	269.49	0.19	51.77	7.01	39.64	7.72	25.29	8.59	16
Bridge	835	12.65	0.10	54.68	0.14	25.24	4.92	20.62	5.51	16.69	5.65	15
Wave	1680	53.76	0.18	398.99	0.28	141.31	9.96	99.61	10.31	78.25	11.08	20
BM Tri	4032	46.27	0.43	184.16	1.58	80.10	28.47	70.56	30.32	56.23	31.09	4
BM Quad	4096	50.71	0.49	277.31	0.847	99.94	31.77	86.05	50.45	66.87	110.80	7
BM Hex	5184	86.90	0.58	368.33	1.25	149.85	33.64	128.83	44.40	108.33	50.88	7
Lilium	7824	102.27	1.56	493.08	2.77	226.72	79.57	178.94	94.71	139.41	92.32	24
Tunnel	10828	46.39	1.99	179.72	3.22	92.77	139.16	75.09	164.93	60.77	157.52	26
Flower	12240	70.33	3.29	363.50	22.30	147.58	433.78	117.54	481.56	92.92	504.78	21
BM 3464	19584	76.77	5.20	411.51	44.36	167.12	272.74	131.11	376.97	100.28	429.39	7
Baku	20768	150.19	7.14	747.46	701.05	352.76	983.55	269.01	1181.69	209.29	1201.44	1
Train Station	23552	119.31	8.20	458.76	935.28	242.22	1433.58	194.69	1536.31	157.23	1646.85	3
Soumaya	24592	355.96	6.37	2520.23	15.45	1318.81	1865.85	845.41	2184.21	544.22	2390.35	25

Table 1. Quantitative analysis of examples (sorted according to the number of beams). For each example, we give the number of beams, computational time and total volume for constructing solutions using 1, 2, 3, 6, and arbitrary types of beams. Increasing the number of beam types reduces the total volume, at the cost of increased manufacturing complexity and slightly more computational time.

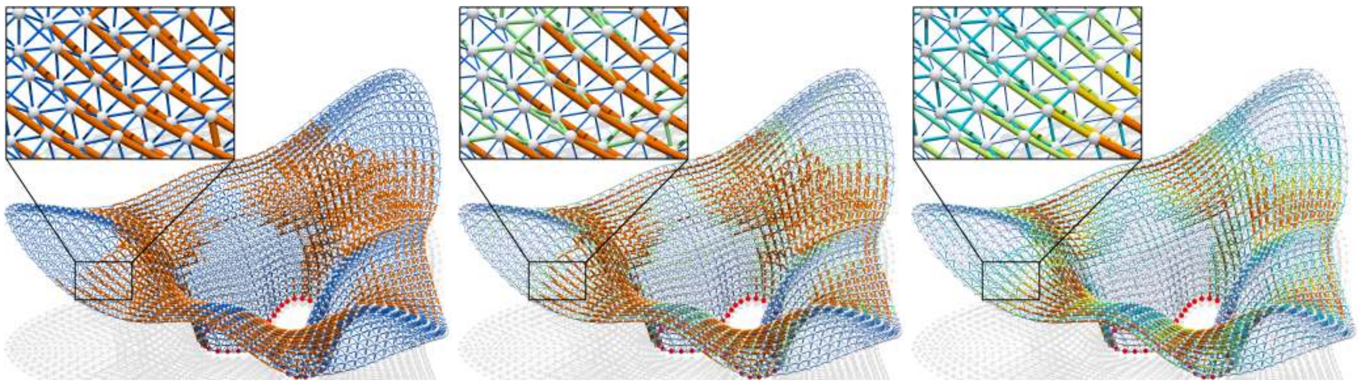


Fig. 21. From left to right: three construction solutions with 2, 3 and 6 types of customized beams for the flower model featuring an extending freeform roof. The thicker beams are adaptively assigned to regions with strong bending moments, as shown in Figure 22.

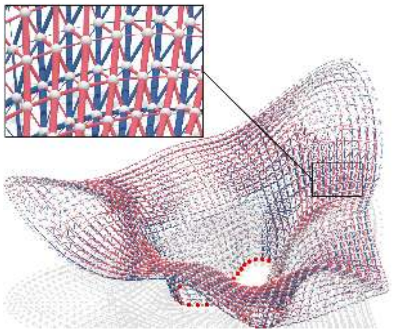


Fig. 22. Color coding of tension (purple) and compression (blue) for the construction solution on the right side of Figure 21. The strong bending moments are transferred to tensile and compressive axial forces along the beams in the two layers.

Structure	Beams	Beam Types	Ours		MILP	
			Vol	T(s)	Vol	T(s)
Small	101	2	15.96	0.83	17.44	0.81
		6	10.19	0.87	10.96	55.20
Dome Fig. 12	600	2	69.59	2.58	70.26	305
		6	50.81	2.64	51.97	620
Lilium Fig. 24	7824	2	226.17	79.57	260.17	610
		6	139.71	92.32	190.16	~ 30h

Table 2. Comparison to mixed integer programming. We show results for three models and two configurations of each model with 2 and 6 beam types. We list the number of beams and compare our volume and running time with the mixed integer programming solutions.

reported in Table 3. The resulting structures are shown in Figure 23 and in the additional materials. The results show that we can reduce the volume up to 20% on average even though our running time is orders of magnitude faster. In addition, previous work can only tackle small structures (e.g., up to 54 bars in [Rasmussen and Stolpe 2008]) while the examples shown in this paper have more than 20,000 bars.

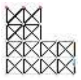
Structure	Stress limit	[Rasmussen and Stolpe 2008]			Ours rounding-up			Ours		
		Vol(m^3)	T	CS($10^{-3}m^2$)	Vol(m^3)	T	CS($10^{-3}m^2$)	Vol(m^3)	T	CS($10^{-3}m^2$)
a 	170Mpa	0.0466	hours	10/5	0.0466	0.4s	10/5	**0.0322**	0.5s	5.30/2.65
	120MPa	0.0608	hours	10/5	0.0608	0.4s	10/5	**0.0456**	0.5	7.5/3.75
	90MPa	0.0608	hours	10/5	0.0608	0.4s	10/5	**0.0608**	0.5	10/5
b	170MPa	0.438	hours	10/7.5/5/2.5	0.455	0.5s	10/7.5/5/2.5	**0.294**	0.6s	4.71/2.04/1.66/0.59
	120MPa	0.524	hours	10/7.5/5/2.5	0.524	0.5s	10/7.5/5/2.5	**0.417**	0.6s	6.67/2.89/2.34/0.83
	90MPa	0.656	hours	10/7.5/5/2.5	0.698	0.5s	10/7.5/5/2.5	**0.555**	0.6s	8.89/3.85/3.14/1.11

Table 3. Computed total volumes for three different methods with the condition of a stress limits of $\pm170\mathbf{MPa}$, $\pm120\mathbf{MPa}$, and $\pm90\mathbf{MPa}$, respectively. (a) a 2D L-shape and (b) a 3D cantilever.

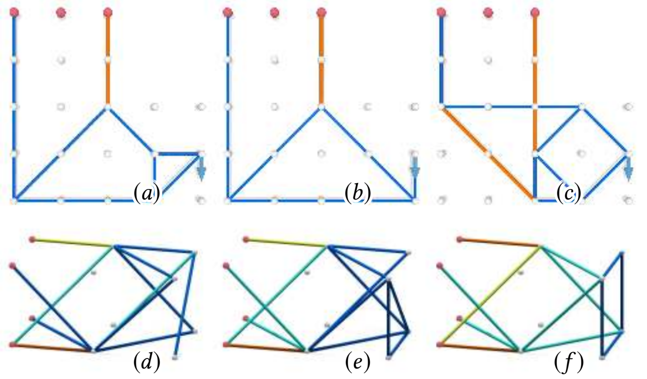


Fig. 23. Computed topologies and assignment of cross sections for three different methods: [Rasmussen and Stolpe 2008](left), our rounding-up method (middle) and our method (right). Top row: a 2D L-shape with a stress limit of $\pm170\mathbf{MPa}$. Bottom row: a 3D cantilever with a stress limit of $\pm90\mathbf{MPa}$.

Alternative Alternating Scheme. A direct extension of [Rasmussen and Stolpe 2008] to accommodate variable cross-section areas is to alternate solving the non-relaxed form of Sp-1 with Sp-3. In Table 2, we compare our approach to this alternative with Mosek as the mixed-integer linear programming (MILP) solver for Sp-1. Our method provides solutions with less volume at a much faster speed and exhibits better scalability. The key problem for mixed integer programming is the initialization. Without a global search strategy, as present in our method, alternating mixed integer programming converges to an undesirable local minimum, in addition to suffering from a much higher computational time.

Discussion and Limitations. Overall, we believe that our algorithm works well, because we coordinated multiple optimization techniques to work well together. We use reformulation, relaxation, splitting and grid search. However, on the theoretical side, we cannot guarantee convergence to a global minimum. This limitation is shared by most other discrete nonlinear optimization algorithms. On the practical side, the main limitation of our work is the lack of control of global buckling. We have not considered the scenario that the overall structure might fail while each beam remains statically sound. In such a case, it is the global structure, as opposed to an individual beam, that might lack stiffness, due to the existence of small eigenvalues in the load-stiffness matrix. This is also a common limitation for other static-aware computational design tools for initial stages, e.g., for self-supporting masonry structures.

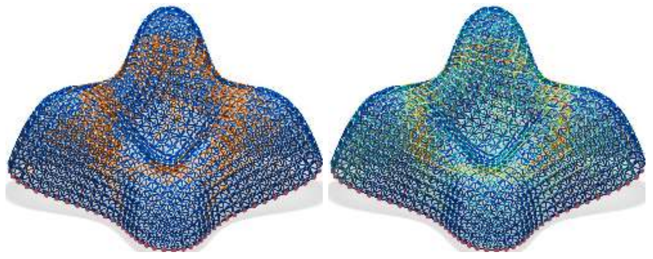


Fig. 24. Construction solutions of a double-layer space structure based on a quadrilateral mesh approximating the Lilium Tower model with 2 (left) and 6 (right) types of customized beams.

6 CONCLUSION AND FUTURE WORK

We study the design and volume optimization of space structures. Our computational framework creates space structures in static equilibrium with reduced material usage. We jointly optimize node positions and connectivity through a nonlinear optimization algorithm based on continuous variables. Moreover, we incorporate the practical consideration of construction complexity in which the types of beams are limited. We solve this challenging problem in a practical and efficient manner. In future work, we plan to study volume optimization under practical considerations for the design of transformable, dynamic, and foldable structures.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments and suggestions for improving the paper. This research was supported by the Visual Computing Center (VCC) at KAUST and by the Max Planck Center for Visual Computing and Communication. Chengcheng Tang would like to thank the support of NSF grant IIS-1528025, a Google Focused Research Award, a gift from the Adobe Corporation, and a hardware donation from NVIDIA. The authors would like to thank Helmut Pottmann, Leonidas Guibas, Renjie Chen, Lorenzo Greco, and Qingyun Sun for discussions, Virginia Unkefer and Olga Diamanti for proofreading, and Marko Tomicic for creating the renderings used in Figure 1 and Figure 3 (middle).

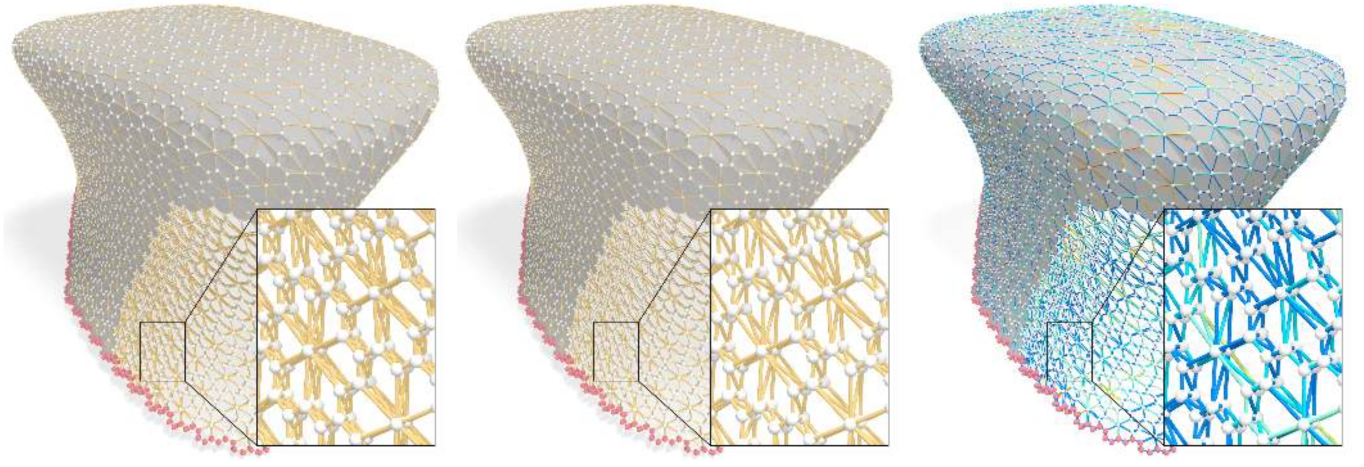


Fig. 25. Soumaya model: An initial configuration contains conflicting beam intersections (left). Joint optimization of node positions and connectivity resolves the conflicts (middle). Discrete optimization further assigns cross-section areas of beams (right).

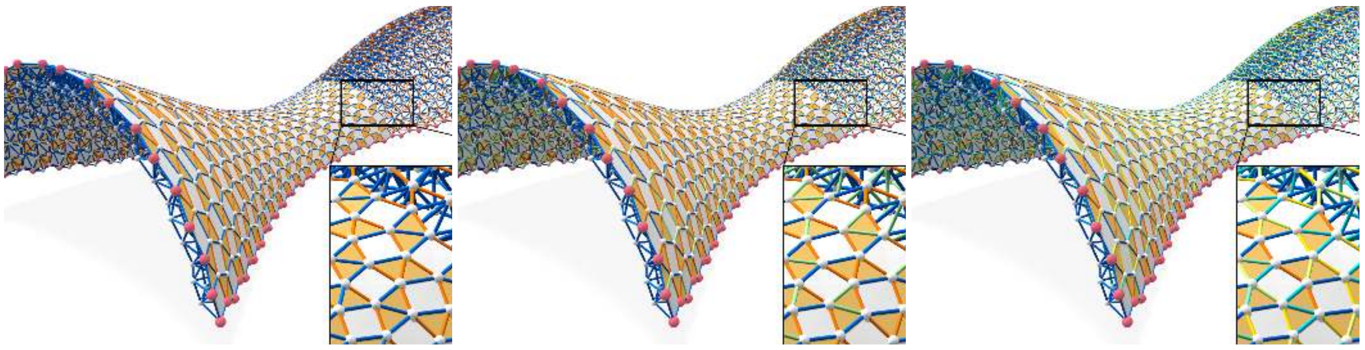


Fig. 26. Construction solutions with 2, 3, and 6 types of beams for a space structure approximating a freeform tunnel. The base mesh is a semi-regular pattern consisting of triangles and quadrilaterals.

REFERENCES

Wolfgang Achtziger and Mathias Stolpe. 2007. Truss topology optimization with discrete design variablesâĂŤguaranteed global optimality and benchmark examples. *Structural and Multidisciplinary Optimization* 34, 1 (2007), 1–20.

Shmuel Agmon. 1954. The relaxation method for linear inequalities. *Canadian Journal of Mathematics* 6 (1954).

MOSEK ApS. 2016. *The MOSEK C optimizer API manual Version 7.1(Revision 62).* http://docs.mosek.com/7.1/capi/index.html

Phillipe Block and John Ochsendorf. 2007. Thrust Network Analysis: A New Methodology For Three-Dimensional Equilibrium. *J. Int. Assoc. Shell and Spatial Structures* 48, 3 (2007), 167–173.

CV Camp and M Farshchin. 2014. Design of space trusses using modified teaching–learning based optimization. *Engineering Structures* 62 (2014), 87–97.

Wai-Fah Chen and Eric M Lui. 2005. *Handbook of Utructural Engineering.* Crc Press.

Fernando de Goes, Pierre Alliez, Houman Owhadi, and Mathieu Desbrun. 2013. On the Equilibrium of Simplicial Masonry Structures. *ACM Trans. Graph.* 32 (2013).

William S Dorn. 1964. Automatic design of optimal structures. *Journal de mecanique* 3 (1964).

Robert M Freund. 2004. Truss design and convex optimization. *Massachusetts Institute of Technologi* (2004).

Chi-Wing Fu, Chi-Fu Lai, Ying He, and Daniel Cohen-Or. 2010. K-set tilable surfaces. *ACM Trans. Graph.* 29 (2010), #44,1–6.

Mathieu Huard, Philippe Bompas, and Michael Eigensatz. 2014. Planar panelization with extreme repetition. In *Advances in Architectural Geometry 2014*, Philippe Block and others (Eds.). Springer.

Caigui Jiang, Chengcheng Tang, Marko Tomičić, Johannes Wallner, and Helmut Pottmann. 2014. Interactive Modeling of Architectural Freeform Structures – Combining Geometry with Fabrication and Statics. In *Advances in Architectural Geometry 2014*, Philippe Block and others (Eds.). Springer.

Caigui Jiang, Chengcheng Tang, Amir Vaxman, Peter Wonka, and Helmut Pottmann. 2015. Polyhedral Patterns. *ACM Trans. Graph.* 34 (2015).

Yoshihiro Kanno and Xu Guo. 2010. A mixed integer programming for robust truss topology optimization with stress constraints. *Internat. J. Numer. Methods Engrg.* 83, 13 (2010), 1675–1699.

A Kaveh, B Farhmand Azar, and S Talatahari. 2008. Ant colony optimization for design of space trusses. *International Journal of Space Structures* 23, 3 (2008), 167–181.

H Kawamura, H Ohmori, and N Kito. 2002. Truss topology optimization by a modified genetic algorithm. *Structural and Multidisciplinary Optimization* 23, 6 (2002), 467–473.

Timothy Langlois, Ariel Shamir, Daniel Dror, Wojciech Matusik, and David IW Levin. 2016. Stochastic structural analysis for context-aware design and fabrication. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 226.

LJ Li, ZB Huang, and F Liu. 2009. A heuristic particle swarm optimization method for truss structures with discrete variables. *Computers & Structures* 87, 7 (2009), 435–443.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming* (1989).

Yang Liu, Hao Pan, John Snyder, Wenping Wang, and Baining Guo. 2013. Computing Self-Supporting Surfaces By Regular Triangulation. *ACM Trans. Graph.* 32 (2013).

Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. 2006. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph.* 25, 3 (2006), 681–689.

Yang Liu, Wenping Wang, Bruno Lévy, Feng Sun, Dong-Ming Yan, Lin Lu, and Chenglei Yang. 2009. On centroidal voronoi tessellationâĂŤenergy smoothness and fast computation. *ACM Trans. Graph.* 28 (2009).

J Clerk Maxwell. 1870. I.âĂŤOn Reciprocal Figures, Frames, and Diagrams of Forces. *Transactions of the Royal Society of Edinburgh* 26, 01 (1870).

Anthony George Maldon Michell. 1904. The limits of economy of material in frame-structures. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 8 (1904).

Daniele Panozzo, Philippe Block, and Olga Sorkine-Hornung. 2013. Designing Unreinforced Masonry Models. *ACM Trans. Graph.* 32 (2013).

Chi-Han Peng, Michael Barton, Caigui Jiang, and Peter Wonka. 2014. Exploring Quadrangulations. *ACM Trans. Graph.* 33 (2014).

Nico Pietroni, Davide Tonelli, Enrico Puppo, Maurizio Froli, Roberto Scopigno, and Paolo Cignoni. 2015. Statics aware grid shells. *Comput. Graph. Forum* 34, 2 (2015), 627–641.

Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. 2013. Make it stand: balancing shapes for 3D fabrication. *ACM Trans. Graph.* 32, 4 (2013), 81.

MH Rasmussen and Mathias Stolpe. 2008. Global optimization of discrete truss topology design problems using a parallel cut-and-branch method. *Computers & Structures* 86, 13 (2008), 1527–1538.

Adriana Schulz, Ariel Shamir, David IW Levin, Pitchaya Sitthi-Amorn, and Wojciech Matusik. 2014. Design and fabrication by example. *ACM Trans. Graph.* 33, 4 (2014), 62.

Jeffrey Smith, Jessica Hodgins, Irving Oppenheim, and Andrew Witkin. 2002. Creating models of truss structures with optimization. *ACM Trans. Graph.* 21 (2002).

Mathias Stolpe and Krister Svanberg. 2003. Modelling topology optimization problems as linear mixed 0–1 programs. *Internat. J. Numer. Methods Engrg.* 57, 5 (2003), 723–739.

Chengcheng Tang, Pengbo Bo, Johannes Wallner, and Helmut Pottmann. 2016. Interactive design of developable surfaces. *ACM Trans. Graph.* 35, 2 (2016), 12.

Chengcheng Tang, Xiang Sun, Alexandra Gomes, Johannes Wallner, and Helmut Pottmann. 2014. Form-finding with Polyhedral Meshes Made Simple. *ACM Tran. Graph.* 33 (2014).

Nobuyuki Umetani, Takeo Igarashi, and Niloy J Mitra. 2012. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph.* 31, 4 (2012), 86–1.

Etienne Vouga, Mathias Höbinger, Johannes Wallner, and Helmut Pottmann. 2012. Design of self-supporting surfaces. *ACM Trans. Graph.* 31, 4 (2012).

Weiming Wang, Tuanfeng Y Wang, Zhouwang Yang, Ligang Liu, Xin Tong, Weihua Tong, Jiansong Deng, Falai Chen, and Xiuping Liu. 2013. Cost-effective printing of 3D objects with skin-frame structures. *ACM Trans. Graph.* 32, 6 (2013).

Dong-Ming Yan, Bruno Lévy, Yang Liu, Feng Sun, and Wenping Wang. 2009. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. *Comput. Graph. Forum* 28 (2009).

Tomás Zegard and Glaucio H Paulino. 2014. GRAND-Ground structure based topology optimization for arbitrary 2D domains using MATLAB. *Structural and Multidisciplinary Optimization* 50 (2014).

Tomás Zegard and Glaucio H Paulino. 2015. GRAND3-Ground structure based topology optimization for arbitrary 3D domains using MATLAB. *Structural and Multidisciplinary Optimization* 52, 6 (2015), 1161–1184.