Improving Solutions to the Truss Topology Design Problem with Alternating Convex Optimization

Hao Yi Ong Conrad Stansbury

June 4, 2014

1 Introduction

In the design of a mechanical structure, we're interested in finding a general layout of the structure's supporting frame that best supports some anticipated design loads. Often, the design of such supporting frames are addressed in ad-hoc ways, such as by complicated domain-specific heuristics [TTG05, DG01]. Our project targets the problem of designing a truss–a collection of bars rigidly attached together that represents the layout of a mechanical structure–subject to constraints expressing limits on quantities like the total cost of materials. Examples of a truss include a construction crane, a railroad bridge, and the Eiffel tower.

All other things being equal, it is desirable for a truss to be stiff in the sense that it does not deflect much under loads placed on the joints of the truss deemed typical for the truss's operation. Because the truss's deflection is not a scalar quantity, it is common in truss optimization problems to use the elastic stored energy of the truss deflected under a particular set of load forces as a proxy for the stiffness, as we do in this paper [Ric13].

Given a graph whose edges are bars in the truss and whose nodes correspond to fixed bar attachment points in physical space, one optimization problem that can reasonably be formed is to find bar cross-sectional areas so that under some set of loads the elastic stored energy is minimized. In the literature this class of problems is referred to as truss topology design (TTD), so called because it aims to optimize a truss with a certain basic graph connectivity (topology) [Fre04].

One shortcoming with solving the TTD problem is that in this method the locations of the nodes and thus those of the bars are fixed. Not only do these solutions look unnatural compared to structures used in practice they are also poorer structures under the metric of elastic stored energy when we compare them with arbitrary trusses with similar topology. In this paper we describe a method using alternating convex programming to locally solve the TTD problem where in addition to solving for the bar cross-sectional areas, we solve for displacements to the original node positions. This way, our algorithm iteratively improves upon a original truss topology as an approximation to optimizing the mechanical structure.

2 Truss Topology Design

In a TTD problem, we are given a base truss topology consisting of an initial set of node positions and the bars to which they are attached to. Our design variables are the bar cross-sectional areas $a \in \mathbf{R}^m$ and the node positions $x \in \mathbf{R}^{2n}$, where $a_i \in \mathbf{R}$ is the area of the i^{th} bar and $x_j \in \mathbf{R}^2$ are the coordinates of the j^{th} node.

We are also given a set of forces anticipating the loads of the actual mechanical structure $F \in \mathbf{R}^{2n}$, where $F_j \in \mathbf{R}^2$ is the load on the j^{th} node. For the bars constituting the truss, we are given their Young's moduli $E_1, \ldots, E_m \in \mathbf{R}$, which characterize the bars' material elasticities.

From the base truss's node positions, we can calculate a force mapping matrix $P \in \mathbf{R}^{m \times 2n}$ with columns $p_1, \ldots, p_{2n}$, which relate the loading forces F to the internal stresses experienced by the bars, $f \in \mathbf{R}^m$; *i.e.*, $Pf + F = 0$. We can also compute the bar lengths $L \in \mathbf{R}^m$, where $L_i = \|x_{i_1} - x_{i_2}\|_2$ is the length of the i^{th} bar connecting nodes at positions x_{i_1} and x_{i_2}.

The truss's deflections are described by $u \in \mathbf{R}^2$, where $u_j \in \mathbf{R}^2$ is the deflection of the j^{th} node. The deflections are related to the internal stresses $f_1, \ldots, f_m \in \mathbf{R}$ by Euler's beam equations

$$f_i = -\frac{E_i a_i}{L_i^2} p_i^T u, \quad i = 1, \ldots, m.$$

The truss deflections are also related to loads F by Hooke's Law $F = Ku$, where

$$K = \sum_{i=1}^{m} \frac{E_i a_i}{L_i^2} p_i p_i^T$$

is the stiffness matrix. We note here that the stiffness matrix is dependent on the bar cross sections a and, through the bar lengths L, the node coordinates x. We highlight these dependencies by using referring to the stiffness matrix as $K(a, x)$ henceforth.

Finally, the stored elastic energy is related to the forces applied on the truss and the node deflections $\Theta = \frac{1}{2} F^T u$, which we minimize in order to maximize the truss stiffness. The optimization problem can thus be stated as follows: Given a set of loading forces $\mathcal{F}$ that the truss is designed to support, a set of fixed nodes $\mathcal{X}^{\text{fixed}}$ indicating where the truss is fixed to the ground, and the physical space $\mathcal{D}$ where the nodes can be placed, the goal is to produce a set of sized bars (*i.e.*, cross-section areas) a and node positions x to minimize the stored elastic energy. Mathematically, we can express the problem as

$$\begin{array}{ll} \text{minimize} & \Theta = \dfrac{1}{2} F^T u \\ \text{subject to} & F = K(a, x) u \\ & a \in \mathcal{A} \\ & x \in \mathcal{X}, \end{array} \tag{1}$$

where $a \in \mathcal{A}$ and $a \in \mathcal{X}$ encode additional design constraints such as truss symmetry and weight limits, based on allowable (convex) cross section areas $\mathcal{A}$ and node coordinates $\mathcal{X}$.

3 An Alternating Convex Optimization Approach

Unfortunately, the minimization of Θ in the space of bar cross-section areas a and node positions x is a non-convex problem. However, the problem can be made locally convex in a and x separately. Thus, as a heuristic to approximately solve (1), we can alternately optimize over the bar sizes a and the node coordinates x. We refer to each pair of bar sizing and node coordinates optimization as an iteration.

3.1 Bar sizing optimization problem

In the bar sizing optimization problem, we optimize the bar cross section areas a (*i.e.*, the design variable) given the vector of node positions x as a constant. To cast the problem formulated as a second-order cone program (SOCP), we introduce a linear change of variables $w, v \in \mathbf{R}^m$:

$$\begin{aligned} w_i + v_i &= -\frac{1}{2}\left(u^T P\right)_i f_i, \quad i = 1, \ldots, m \\ w_i - v_i &= a_i, \quad i = 1, \ldots, m. \end{aligned} \tag{2}$$

Intuitively, the value of $w_i + v_i$ is the stored elastic energy in the i^{th} bar.

With (2) and the relations encoded by the problem data given in §2, we can cast (1) with x and its corresponding substitutions held constant as an SOCP:

$$\begin{array}{ll} \text{minimize} & \Theta = 1^T (w + v) \\ \text{subject to} & Pf + F = 0 \\ & \left\| \left(v_i, \dfrac{L_i}{\sqrt{E_i}} f_i \right) \right\|_2 \leq w_i, \quad i = 1, \ldots, m \\ & w \in \mathcal{W} \\ & v \in \mathcal{V}, \end{array} \tag{3}$$

where $w \in \mathcal{W}$ and $v \in \mathcal{V}$ enforce the additional design constraints originally encoded by $a \in \mathcal{A}$ and $a \in \mathcal{X}$ using convex sets $\mathcal{W}$ and $\mathcal{V}$.

3.2 Node coordinates optimization problem

In the node coordinates optimization problem, we optimize the node positions x given the vector of bar sizes a as a constant. Instead of directly optimizing over x, however, we optimize over its displacement, $y \in \mathbf{R}^n$, where y_j is the displacement of the j^{th} node; *i.e.*, we have the update $x^+ = x + y$. As it stands, the problem is still not convex in y. This is due to the way that $x + y$ is related through the internal stresses $f_1, \ldots, f_m$ to the node deflections variable u. Under the assumption that y is small, however, it is reasonable to Taylor expand around $y = 0$ to obtain

$$f_i = -\frac{E_i a_i}{L_j^2}\left(1 + \frac{2p_i^T y}{L_j} p_i p_i^T\right), \tag{4}$$

which allows us to formulate the node coordinates optimization problem as a convex one.

Analogous to the bar sizing optimization problem, we introduce an affine change of variables $w, v \in \mathbf{R}^m$ (different from (2)) to cast the optimization problem as an SOCP:

$$\begin{aligned} w_i + v_i = &\quad -\frac{1}{2}\left(u^T P\right)_i f_i, \quad i = 1, \ldots, m \\ w_i - v_i = &\quad \frac{2 p_i y_i}{L_i} + 1, \quad i = 1, \ldots, m. \end{aligned} \tag{5}$$

With (4), (5), and the relations encoded by the problem data given in §2, we can formulate (1) with a and its corresponding substitutions held constant as an SOCP:

$$\begin{array}{ll} \text{minimize} & \Theta = 1^T (w + v) \\ \text{subject to} & Pf + F = 0 \\ & \frac{1}{2}\left((w_i - vi) - 1\right) = \frac{p_i^T y_i}{L_i}, \quad i = 1, \ldots, m \\ & \left\| \left(v_i, \frac{L_i}{\sqrt{E_i a_i}} f_i \right) \right\|_2 \leq w_i, \quad i = 1, \ldots, m \\ & w \in \mathcal{W} \\ & v \in \mathcal{V} \\ & y \in \mathcal{Y}, \end{array} \tag{6}$$

where, again, $w \in \mathcal{W}$ and $v \in \mathcal{V}$ enforce the additional design constraints originally encoded by $a \in \mathcal{A}$ and $a \in \mathcal{X}$ using convex sets $\mathcal{W}$ and $\mathcal{V}$. Additionally, we include $y \in \mathcal{Y}$, which enforces a set of convex constraints such as truss symmetry and node displacement magnitude bounds (node migration rate ϵ) to ensure the validity of our Taylor expansion.

3.3 Alternating convex optimization algorithm

With our problem data and two optimization subproblems defined, we can thus describe our algorithm as follows:

Algorithm 1: *Alternating convex optimization method for TTD*

given a set of fixed nodes $\mathcal{X}^{\text{fixed}}$, a set of force loadings $\mathcal{F}$, and an allowable physical space $\mathcal{D}$.

Generate initial set of discretized node coordinates x^0 from $\mathcal{D}$, set $x := x^0$.

repeat

1. Given x, obtain a and Θ_1–the solution to and objective of (3).
2. Given a, obtain y and Θ_2–the solution to and objective of (6).
3. Update $x := x + y$.
4. **break if** Θ_1 and Θ_2 converge.

return a, x.

---

4 Example: Bridge Design

To test and demonstrate our method, we applied it to the problem of designing a 2-D bridge with two fixed nodes and a design force evenly distributed between them. To generate a topology to optimize, we placed the fixed nodes in the left and right bottom corners of a uniform unit lattice with $n_x \times n_y$ nodes.

As a constraint on the lengths of bars, we limit our edge set to edges that only connect pairs of nodes a distance no more than 3 units apart. This constraint is subsumed in $a \in \mathcal{A}$ such that the cross sections of longer bars are constrained to 0. We also placed a total volume constraint and a maximum area constraint on the bars

$$L^T a \leq V^{\max}$$

$$a \preceq a^{\max}\mathbf{1},$$

as well as a limit on the total node migration distance from the original truss topology $\rho = 0.4$ units on each node's displacement to prevent nodes from passing each other and invalidating our Taylor expansion (4). In our example we used a node migration rate limit $\epsilon = 0.015$ units that limits the magnitudes of node displacement y in each iteration. These values were determined experimentally to provide a good trade-off between the validity of our approximations and the number of iterations required for adequate convergence.

We present the results of applying our method to a problem with $n_x = 8, n_y = 4$ in Figures 1 and 2.

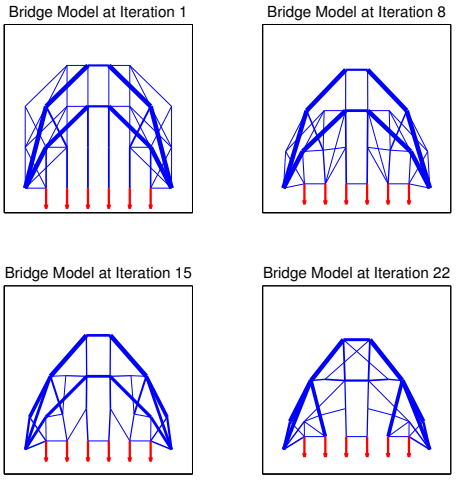

Figure 1: Line thickness indicates bar cross-sections. Red arrows are the loading forces for the problem.

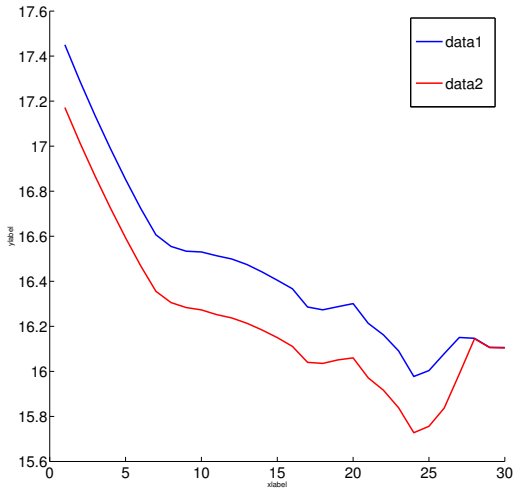

Figure 2: Objective values throughout run for the model in Figure 1. Optimal structure obtained at iteration 24.

The upper left inset in Figure 1 shows the result after one "half-iteration" (finding the cross-sectional areas), which is the solution to the standard TTD problem. The lower right

inset shows the result at convergence. Because the problem is non-convex, the solution at practical convergence was not the best structure, as seen in Figure 2.

Our solution has a reduced number of bars without our objective containing any regularization terms, which is another positive characteristic of a truss with all else being equal. We also note that our solution has a single arch with a classic inverted catenary shape. For this model the elastic stored energy was reduced by 8.9%. We also note that for this model, the constraints we impose seem to imply convergence in a finite number of iterations.

5 Implementation and Analysis

Because both of the convex subproblems (3) and (6) are SOCPs it was possible to test our method on trusses of substantial size using the interior point solvers distributed with CVX, the Matlab convex modeling framework. Using CVX, we were also able to submit our problem to SCS, a convex cone solver which uses operator splitting [OCPB13].

We found that SCS solved our problems significantly faster than SeDuMi, even without providing warm starts to the solver. A summary of these results for a few different problem sizes with the bridge topology can be seen in Table 1 in Appendix A. While we encountered a few issues in using SCS relating to numerical stability, these were avoided by choosing a tolerance of 10^{-9}. This made the duality gap parameter comparably small to those of the interior point solvers.

6 Conclusions and Future Work

We present a promising approach to improve upon solutions to the truss design problem. More work should be done to investigate convergence over a variety of models. From an engineering standpoint, it might be interesting if real structures were used as the original truss topology upon which our algorithm improves. With regards to the optimization procedure, we are interested in seeing how to apply techniques from distributed optimization to the problem, though in the time being the results from SCS are satisfying on this front.

Acknowledgments

We thank Professor Stephen Boyd and the course instructors for their motivating this project and giving feedback. We are also grateful to the peer review teams who gave input on our proposal and midterm report. Finally, we thank Brendan O'Donoghue for promptly patching the convex cone solver SCS, which enabled our numerical experiments with this solver.

References

[DG01] K. Deb and S. Gulati. Design of truss-structures for minimum weight using genetic algorithms. *Finite Elements in Analysis and Design*, 37:447–465, 2001.

[Fre04] R. M. Freund. Truss design and convex optimization. 2004.

[OCPB13] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. Operating splitting for conic optimization via homogeneous self-dual embedding. 2013.

[Ric13] J. N. Richardson. *Topology Optimization of Truss-like Structures: from Theory to Practice*. PhD thesis, Brussels School of Engineering, 2013.

[TTG05] W. Tang, L. Tong, and Y. Gu. Improved genetic algorithm for design optimization of truss structures with sizing, shape and topology variables. *International Journal for Numerical Methods in Engineering*, 62:1737–1762, 2005.

A Simulation results for bridge design examples

	small: 8×4	**medium:** 20×10	**large:** 30×15
std. form variables (CSP)	791	6701	16026
std. form constraints	219	1737	4102
SeDuMi solve time	**0.712 sec**	**11.40 sec**	**112.8 sec**
SCS solve time	**0.310 sec**	**2.496 sec**	**7.368 sec**
SCS iterations per solve	8080	14220	10080
std. form variables (NMP)	1114	8530	19990
std. form constraints	464	3338	7713
SeDuMi solve time	**1.287 sec**	**25.82 sec**	**259.4 sec**
SCS solve time	**0.280 sec**	**1.320 sec**	**5.240 sec**
SCS iterations per solve	1120	1580	1540

Table 1: SCS and SeDuMi results on the bridge model. CSP refers to the cross-sections subproblem (3), and NMP the node migrations subproblem (6). Times are per iteration on an i7 2720QM.