An Integrated Approach for Truss Structure Design

John N. Hooker[1] Tallys Yunes[2]

1 Introduction

Truss structure optimization is one of the fundamental problems in engineering design. The goal in this mixed discrete/nonlinear problem is to find a minimum-cost placement and sizing of truss members (bars) to support a given load. The possible cross-sectional areas of the bars come from a discrete set of values, which correspond to commonly manufactured sizes. The structure to be built is represented as a network of nodes and arcs in two or three dimensions. The coordinates of the nodes are given, and structural bars are joined at the nodes. The problem consists of selecting the cross-sectional area of the bar to be placed along each arc, where the area is zero if no bar is placed. The objective is to minimize cost, which in our case is the volume of metal (aluminum) used in the bars. Each node has freedom of movement in a specified number of directions (degrees of freedom). Several possible loading conditions are anticipated, each of which is represented by a force applied to a subset of the nodes and along one or more degrees of freedom. There are limits on the displacement of each node along their degrees of freedom for each loading condition. In addition, the elongation, compression and stress on each bar must lie within given limits.

Due to the presence of nonconvex physical constraints coming from Hooke's law, these problems are very difficult to solve. The initial nonlinear (MINLP) formulation can be solved directly with a global optimization solver (such as BARON, Tawarmalani and Sahinidis 2005), or converted into an MILP model by adding 0-1 and continuous variables (Ghattas and Grossmann 1991). A third approach, which was implemented by Bollapragada, Ghattas and Hooker (2001) is to use a special kind of relaxation known as a *quasi-relaxation*, paired with logic cuts and an effective branching scheme. We describe an integrated model written and solved with SIMPL that mimics this latter approach with improved results. SIMPL (Yunes, Aron and Hooker 2009) is a modeling and solution system specifically designed to facilitate the creation of high-level integrated models that take advantage of low-level cooperation between different solution techniques such as constraint programming and integer programming.

2 Alternative Problem Formulations

The problem data consists of the following: I is the number of bars, J is the number of degrees of freedom (summed over all nodes), L is the number of loading conditions, and K_i is the number of discrete cross-sectional areas for bar i. Also, h_i is the length of bar i, A_{ik} is the k-th discrete cross sectional area of bar i, E_i is the modulus of elasticity of bar i, $p_{j\ell}$ is the force imposed by loading condition ℓ at degree of freedom j, b_{ij} is the cosine of the angle between bar i and degree of freedom j, and c_i is the cost per unit volume of bar i (typically the weight density). Finally, σ_i^L and σ_i^U are the minimum and maximum allowable stress in bar i, v_i^L and v_i^U are the limits on elongation/contraction of bar i, and d_j^L and d_j^U are the displacement limits for degree of freedom j.

[1]Carnegie Mellon University, Pittsburgh, PA 15213-3890, john@hooker.tepper.cmu.edu
[2]University of Miami, Coral Gables, FL 33124-8237, tallys@miami.edu

An MINLP formulation of the problem is given by

$$
\begin{aligned}
&\min \sum_{i=1}^{I} c_i h_i A_i \\
&\sum_{i=1}^{I} b_{ij} s_{i\ell} = p_{j\ell}, \text{ all } j,\ell && \text{(a)} \\
&\sum_{j=1}^{J} b_{ij} d_{j\ell} = v_{i\ell}, \text{ all } i,\ell && \text{(b)} \\
&\frac{E_i}{h_i} A_i v_{i\ell} = s_{i\ell}, \text{ all } i,\ell && \text{(c)} \\
&v_i^L \le v_{i\ell} \le v_i^U, \text{ all } i,\ell && \text{(d)} \\
&d_j^L \le d_{j\ell} \le d_j^U, \text{ all } j,\ell && \text{(e)} \\
&\bigvee_{k=1}^{K_i} (A_i = A_{ik}), \text{ all } i && \text{(f)}
\end{aligned}
\tag{1}
$$

The variables are as follows. A_i is the cross sectional area chosen for bar i, where the absence of bar i is represented by assigning a very small value to A_i; $s_{i\ell}$ is the force in bar i due to loading condition ℓ; $\sigma_{i\ell}$ is the stress in bar i due to loading condition ℓ; $v_{i\ell}$ is the elongation (or contraction, if negative) of bar i due to loading condition ℓ; $d_{j\ell}$ is the node displacement along degree of freedom j for loading condition ℓ. Constraints (a) are equilibrium equations that balance the external loads with the forces induced in the bars; (b) are compatibility equations that relate the displacement of the nodes with the elongation of the bars; (c) represent Hooke's law, which relates the elongation or compression of a bar to the force applied to it (note that this constraint is nonlinear); and the disjunctive constraints (f) require that each area A_i take one of the discrete values A_{ik}.

If we add 0-1 variables y_{ik} that are equal to 1 when $A_i = A_{ik}$, (f) can be replaced by two constraints, $A_i = \sum_k A_{ik} y_{ik}$ and $\sum_k y_{ik} = 1$, for each i. This transforms (1) into a mixed-integer nonlinear programming model (MINLP), which can be solved by a global optimization solver like BARON. Furthermore, if we disaggregate the $v_{i\ell}$ variables, we can convert (1) to an MILP model by replacing $v_{i\ell}$ with $\sum_k v_{ik\ell}$ and replacing (c) with $\frac{E_i}{h_i} \sum_{k=1}^{K_i} A_{ik} v_{ik\ell} = s_{i\ell}$, for all i,ℓ. A disadvantage of the MILP model is the large number of variables. Bollapragada, Ghattas and Hooker (2001) show how to get a much smaller relaxation of a problem that has the same optimal value as (1). The resulting *quasi-relaxation* of (1) therefore provides a valid bound on the optimal value of (1).

The quasi-relaxation technique, generalized in Hooker (2005), applies to any constraint of the form $g(x,y) \le 0$ where g is semihomogeneous in x and concave in y, and where $x \in \mathbb{R}^n$ and y is a scalar. The function $g(x,y)$ is semihomogeneous in x when $g(\alpha x, y) \le \alpha g(x,y)$ for all x, y and $\alpha \in [0,1]$ and $g(0,y) = 0$ for all y. We also suppose there are bounds $x^L \le x \le x^U$ and $y_L \le y \le y_U$, and the objective function involves variables in x. Then a quasi-relaxation can be obtained by decomposing x with the constraint $x = x^1 + x^2$, and replacing $g(x,y) \le 0$ with

$$
\begin{aligned}
&g(x^1, y_L) + g(x^2, y_U) \le 0 \\
&\alpha x^L \le x^1 \le \alpha x^U \\
&(1-\alpha) x^L \le x^2 \le (1-\alpha) x^U
\end{aligned}
$$

This idea is sufficiently general to justify a metaconstraint that represents constraints of the form $g(x, y) \leq 0$ satisfying the above conditions. (The bilinear constraints (c) in (1) satisfy these conditions.) In SIMPL, constraints are the central elements that control search, inference and relaxation. We use the term *metaconstraint* to refer to a constraint endowed with additional attributes and/or operations, such as: how to relax itself, how to branch on itself, and how to infer other constraints from itself. The SIMPL model of Section 3 will invoke the quasi-relaxation simply by adding an additional relaxation option to a bilinear metaconstraint representing (c) in (1).

A quasi-relaxation of (1) is written as follows:

$$
\begin{aligned}
&\min \sum_{i=1}^{I} c_i h_i \left[A_i^L y_i + A_i^U (1 - y_i)\right] \\
&\sum_{i=1}^{I} b_{ij} s_{i\ell} = p_{j\ell}, \text{ all } j, \ell \\
&\sum_{j=1}^{J} b_{ij} d_{j\ell} = v_{i0\ell} + v_{i1\ell}, \text{ all } i, \ell \\
&\frac{E_i}{h_i} \left(A_i^L v_{i0\ell} + A_i^U v_{i1l}\right) = s_{i\ell}, \text{ all } i, \ell \\
&v_i^L y_i \leq v_{i0\ell} \leq v_i^U y_i, \;\; v_i^L (1 - y_i) \leq v_{i1\ell} \leq v_i^U (1 - y_i), \text{ all } i, \ell \\
&d_j^L \leq d_{j\ell} \leq d_j^U, \text{ all } j, \ell \quad \text{and} \quad 0 \leq y_i \leq 1, \text{ all } i
\end{aligned}
\tag{2}
$$

Model (1) can be solved by branch-and-bound using (2) as the relaxation at each node. If $0 < y_i < 1$ in the optimal solution of (2) and $A_i^L \neq A_i^U$, let $A_i^* = A_i^L y_i + A_i^U (1 - y_i)$. We split the domain of A_i into two parts: $A_i \in \{\text{cross-sectional areas} < A_i^*\}$, and $A_i \in \{\text{cross-sectional areas} \geq A_i^*\}$. We branch first on the second part because it is more likely to lead to a feasible solution. Another property of (2) is that, whenever it is feasible, there exists a feasible solution in which both $v_{i0\ell}$ and $v_{i1\ell}$ have the same sign. Therefore, if $v_{i0\ell}$ and $v_{i1\ell}$ have opposite signs in the solution of (2), we can branch by introducing *logic cuts* of the form $v_{i0\ell}, v_{i1\ell} \geq 0$ (left branch), and $v_{i0\ell}, v_{i1\ell} \leq 0$ (right branch). These logic cuts are particularly useful in the presence of displacement bounds ($d_j^L > -\infty$ and $d_j^U < \infty$), and they are checked for violation before the check on y_i.

3 Integrated Approach

We now describe the SIMPL model for (1). Note that constraints (d), (e) and (f) are part of the variable declarations (the data and variable declaration sections are omitted).

```
01. OBJECTIVE
02.   maximize sum i of c[i]*h[i]*A[i]
03. CONSTRAINTS
04.   equilibrium means {
05.     sum i of b[i,j]*s[i,l] = p[j,l] forall j,l
06.     relaxation = { lp } }
07.   compatibility means {
08.     sum j of b[i,j]*d[j,l] = v[i,l] forall i,l
09.     relaxation = { lp } }
```

```
10.   hooke means {
11.     E[i]/h[i]*A[i]*v[i,l] = s[i,l] forall i,l
12.     relaxation = { lp:quasi } }
13. SEARCH
14.   type = { bb:bestdive }
15.   branching = { hooke:first:quasicut, A:splitup }
```

Inside SIMPL, each constraint in line 11 becomes two constraints: `E[i]/h[i]*z[i,l] = s[i,l]` and `bilinear(A[i],v[i,l],z[i,l])` (which imposes $A_i v_{i\ell} = z_{i\ell}$ among other things). The `lp:quasi` statement creates a quasi-relaxation of the `bilinear` constraint and sends it to the LP solver. This relaxation is volatile (i.e. it is updated whenever the lower and/or upper bounds on A_i change). We first branch by looking for the first violation of the logic cuts (`hooke:first:quasicut`), and then by choosing the first A_i that violates its indomain constraint. Following a recommendation by Bollapragada, Ghattas and Hooker (2001), we turn off logic cuts when the problem has no displacement bounds (i.e. when $d_j^L = -\infty$ and $d_j^U = \infty$). When no logic cuts are found/enabled, branching is performed on the A_i variables. The domain of the chosen A_i is split at the current fractional value and we branch first on the upper (right) piece (`splitup`). The problem also includes the concept of equivalence classes (or linking groups), which are subsets of bars that are supposed to have the same cross-sectional area (a requirement of the application). The choice of the A_i variable on which to branch can be prioritized to scan the bars in non-increasing order of linking group size. To do this, we can modify the `A:splitup` statement in line 15 to `A:most(1,LinkSize):splitup`. Here, `LinkSize` is a vector of numbers (with the same dimension as the `A` vector). The number `1` indicates that the values in this vector are to be used as the first sorting criterion which, as a consequence, makes the violation amount the second sorting criterion.

4 Computational Results

For our computational experiments we used 12 instances from the literature (the same ones used by Bollapragada, Ghattas and Hooker 2001). Eleven of them come from Venkayya (1971), and one of them from Cai and Theirauf (1993). Table 1 shows the number of search nodes and CPU time (in seconds) required to solve these problems with each of the following four approaches: solving the MILP version of model (18) with CPLEX 11; solving the MINLP version of model (18) with BARON; using the original implementation of Bollapragada, Ghattas and Hooker (2001) (referred to as BGH); and replicating the BGH approach with SIMPL. BARON was run on an IBM workstation with two 3.2 GHz Intel Xeon processors and 2.5 GB of RAM. The other three algorithms were run on the same machine as the other experiments in this paper. All runs had a time limit of 24 CPU hours.

As the problem size increases, the standard global optimization approach (BARON column) does not scale well and time becomes a factor. The MILP model solved by CPLEX tends to get too large as the number of bars increases, and the solution time grows more quickly than in the integrated approaches. It is worth noting that CPLEX found a solution of value 5096.99 for instance 3, whereas BARON, BGH and SIMPL all found a solution of value 5156.64 (if we use CPLEX version 9 instead of 11, the "optimal" solution it returns for instance 3 has value 5037.4). Both BGH and SIMPL behave very similarly (as expected), with SIMPL having some advantage on the larger instances. Surprisingly, even though SIMPL's underlying code has considerably more

Table 1: Number of search nodes and CPU time (in seconds). Only CPLEX, BGH and SIMPL were run on the same machine. *CPLEX's solution to instance 3 is apparently incorrect. †Instance 9 was run in SIMPL with depth-first search (like BGH).

		BARON		CPLEX 11		BGH		SIMPL	
Problem	#Bars	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time
1A	10	263	5.26	390	0.40	95	0.03	83	0.08
1B	10	175	3.83	106	0.26	81	0.02	73	0.07
1C	10	479	8.12	702	0.83	521	0.16	533	0.49
1D	10	518	8.76	1,320	1.17	719	0.22	726	0.63
2	10	449	24.28	2,977	4.86	841	0.64	1,028	1.84
3	10	11,354	327.19	403,683*	146.14*	517,255	144.67	94,269	64.75
4	10	34,662	2,067.43	678,471	1,086.72	1,088,955	600.09	508,816	650.71
5	25	3,190	3,301.62	3,739	43.65	11,351	44.09	2,401	20.23
6	72	291	3,375.93	1,962	207.81	665	33.01	489	27.93
7	90	782	21,610.86	2,376	576.46	1,889	130.86	826	92.47
8	108	*no sol.*	$>$ 24h	14,966	3,208.38	9,809	1,996.87	8,485	1,719.99
9	200	*no sol.*	$>$ 24h	*no sol.*	$>$ 24h	*feasible*	$>$ 24h	*feasible*	$>$ 24h
						cost =	32747.58	*cost* =	32700.25†

overhead than the BGH code (which was solely written for the purpose of solving the truss design problem), SIMPL is only about two and a half times slower than BGH (on average, over the 12 instances) in terms of number of nodes processed per second.

For the 200-bar instance, in particular, neither BARON nor CPLEX were able to find a single feasible solution within 24 hours. Both BGH and SIMPL found a feasible (but not provably optimal) solution in less than 24 hours, and the solution found by SIMPL (32700.25) was slightly better than the one found by BGH (32747.58).

References

Bollapragada, S., O. Ghattas, J. N. Hooker. 2001. Optimal design of truss structures by mixed logical and linear programming, *Operations Research* 49(1), 42–51.

Cai, J., G. Thierauf. 1993. Discrete Optimization of structures using an improved penalty function method, *Engineering Optimization* 21, 293–306.

Ghattas, O., I. Grossmann. 1991. MINLP and MILP strategies for discrete sizing structural optimization problems, *Proc. of ASCE 10th Conference on Electronic Computation*, Indianapolis.

Hooker, J. N. 2005. Convex programming methods for global optimization, in C. Jermann, A. Neumaier, and D. Sam, eds., *Global Optimization and Constraint Satisfaction (COCOS 2003 invited talk)*, *Lecture Notes in Computer Science* 3478, 46–60.

Tawarmalani M., N. V. Sahinidis. 2005. A polyhedral branch-and-cut approach to global optimization, *Mathematical Programming* 103(2), 225–249.

Venkayya, V. B. 1971. Design of optimum structures, *Computers and Structures* 1, 265–309.

Yunes, T., I. D. Aron, J. N. Hooker. 2010. An integrated solver for optimization problems, *Operations Research* 58(2), 342–356.