Global Optimization of Truss Structure Design

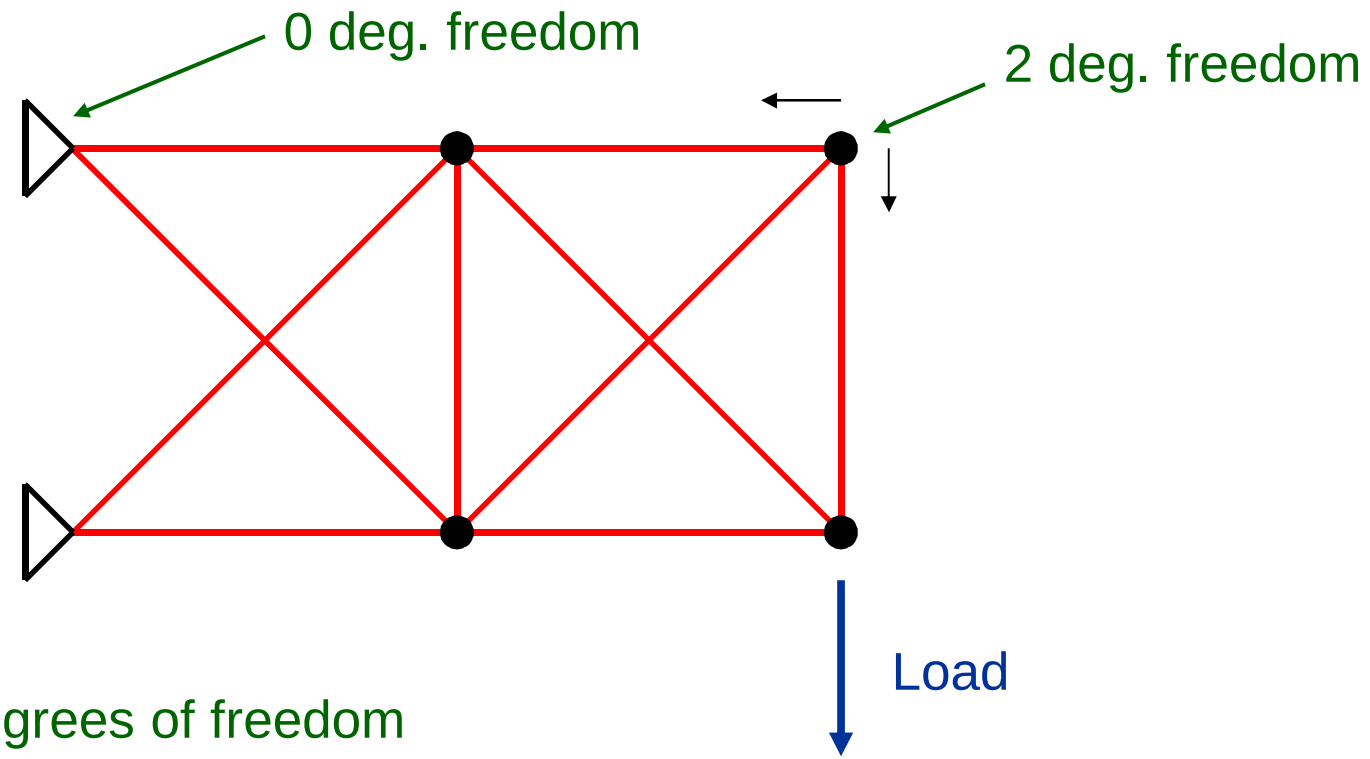
J. N. Hooker
Tallys Yunes

INFORMS 2010

Truss Structure Design

Select size of each bar (possibly zero) to support the load while minimizing weight. Bar sizes are **discrete**.

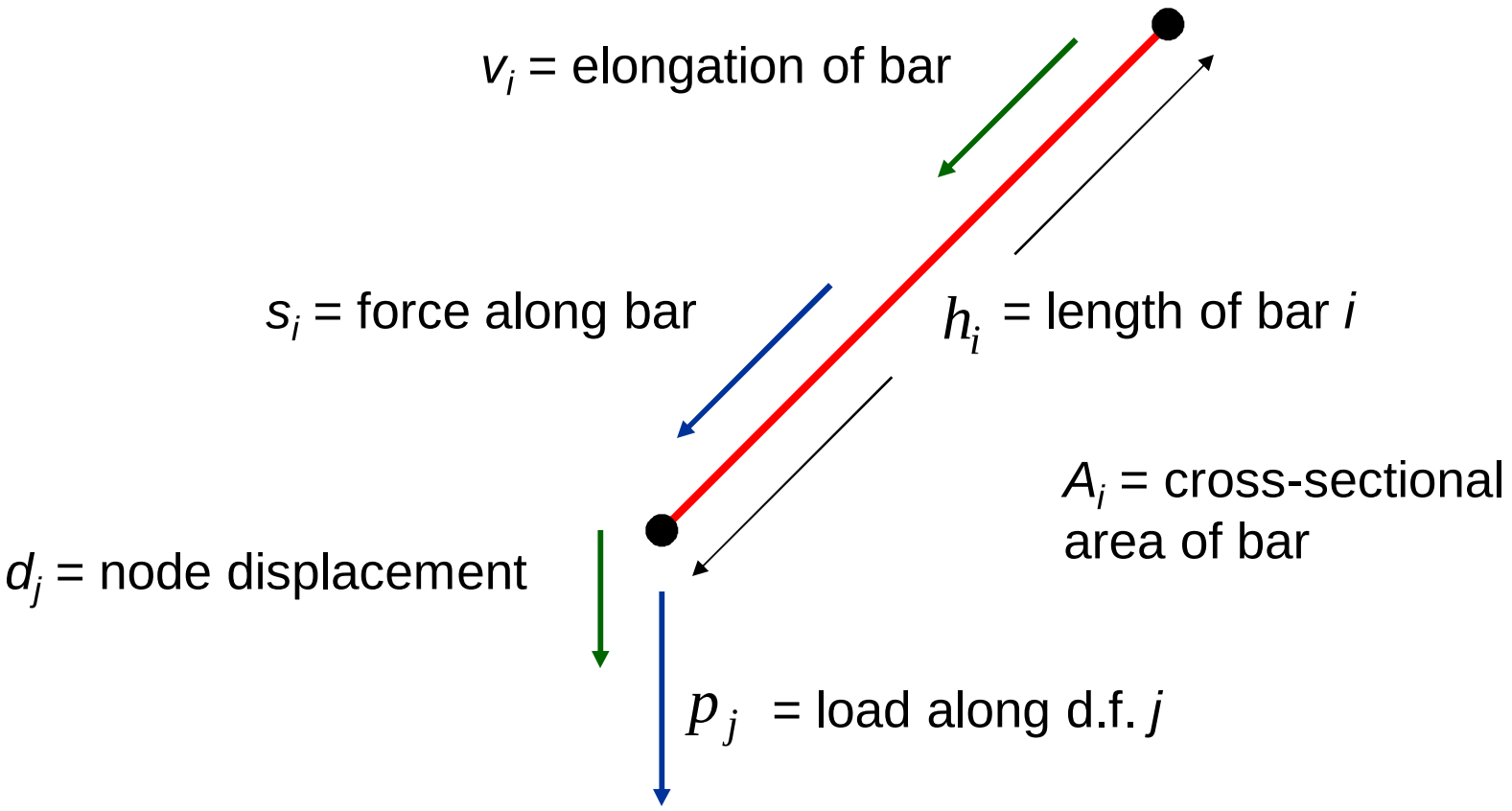
10-bar cantilever truss



Total 8 degrees of freedom

Truss Structure Design

Notation



Truss Structure Design

$$\min \quad \sum_i h_i A_i$$ } Minimize total weight

$$s.t. \quad \sum_i \cos\theta_{ij} s_i = p_j, \text{ all } j$$ } Equilibrium

$$\sum_j \cos\theta_{ij} d_j = v_i, \text{ all } i$$ } Compatibility

nonlinear →

$$\frac{E_i}{h_i} A_i v_i = s_i, \text{ all } i$$ } Hooke's law

$$v_i^L \le v_i \le v_i^U, \text{ all } i$$ } Elongation bounds

$$d_j^L \le d_j \le d_j^U, \text{ all } j$$ } Displacement bounds

$$\bigvee_k (A_i = A_{ik})$$ } Logical disjunction

Area must be one of several discrete values A_{ik}

Constraints can be imposed for multiple loading conditions

Truss Structure Design

Introducing new variables linearizes the problem but makes it much larger.

MILP model

$$\min \quad \sum_i h_i \sum_k A_{ik} y_{ik}$$

$$s.t. \quad \sum_i \cos\theta_{ij} s_i = p_j, \text{ all } j$$

$$\sum_j \cos\theta_{ij} d_j = \sum_k v_{ik}, \text{ all } i$$

$$\frac{E_i}{h_i} \sum_k A_{ik} v_{ik} = s_i, \text{ all } i$$

$$v_i^L \le v_i \le v_i^U, \text{ all } i$$

$$d_j^L \le d_j \le d_j^U, \text{ all } j$$

$$\sum_k y_{ik} = 1, \text{ all } i$$

0-1 variables indicating size of bar *i*

Elongation variable disaggregated by bar size

Hooke's law becomes linear

Solution Approach

- Solve with **SIMPL**, an integrated solver for optimization.
- SIMPL integrates MILP, constraint programming, global optimization in a **unified** approach.

SIMPL Integration Principles

- **Low-level** integration with **high-level** modeling.

SIMPL Integration Principles

- **Low-level** integration with **high-level** modeling.
- Succinct modeling with **meta-constraints**.
  - Model communicates **problem structure** to the solver.

SIMPL Integration Principles

- **Low-level** integration with **high-level** modeling.
- Succinct modeling with **meta-constraints**.
  - Model communicates **problem structure** to the solver.
- General **search-infer-relax** solution algorithm.
  - Enumerate problem restrictions.
  - Branching or logic-based Benders.
  - Underlying search/inference and search/relaxation dualities.

SIMPL Integration Principles

- **Low-level** integration with **high-level** modeling.
- Succinct modeling with **meta-constraints**.
  - Model communicates **problem structure** to the solver.
- General **search-infer-relax** solution algorithm.
  - Enumerate problem restrictions.
  - Branching or logic-based Benders.
  - Underlying search/inference and search/relaxation dualities
- **Constraint-based** control.
  - Filtering, relaxation, branching.

Classical solution methods

- CP solver
  - **Search:** Branching
  - **Inference:** Flltering
  - **Relaxation:** Domain store
- MILP solver
  - **Search:** Branching
  - **Inference:** Cutting planes, presolve, reduced cost variable fixing
  - **Relaxation:** LP
- Benders
  - **Search:** Enumerate subproblems.
  - **Inference:** Benders cuts
  - **Relaxation:** Master problem

Classical solution methods

- Global optimization
  - **Search:** Enumerate boxes
  - **Inference:** Domain reduction, dual-based variable bounding
  - **Relaxation:** Convexification
- SAT
  - **Search:** Branching
  - **Inference:** Conflict clauses
  - **Relaxation:** Same as restriction
- Local search
  - **Search:** Enumerate neighborhoods.
  - **Inference:** Tabu list, etc.
  - **Relaxation:** Same as restriction

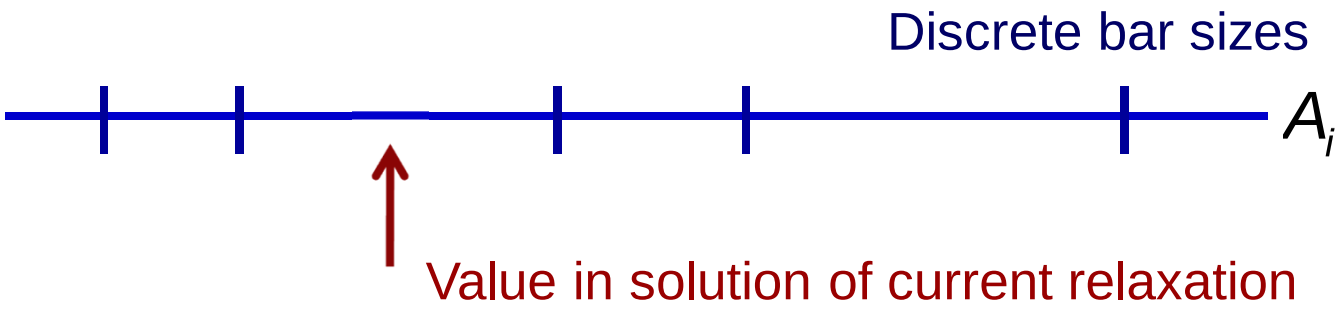
Truss Structure Design

Integrated approach

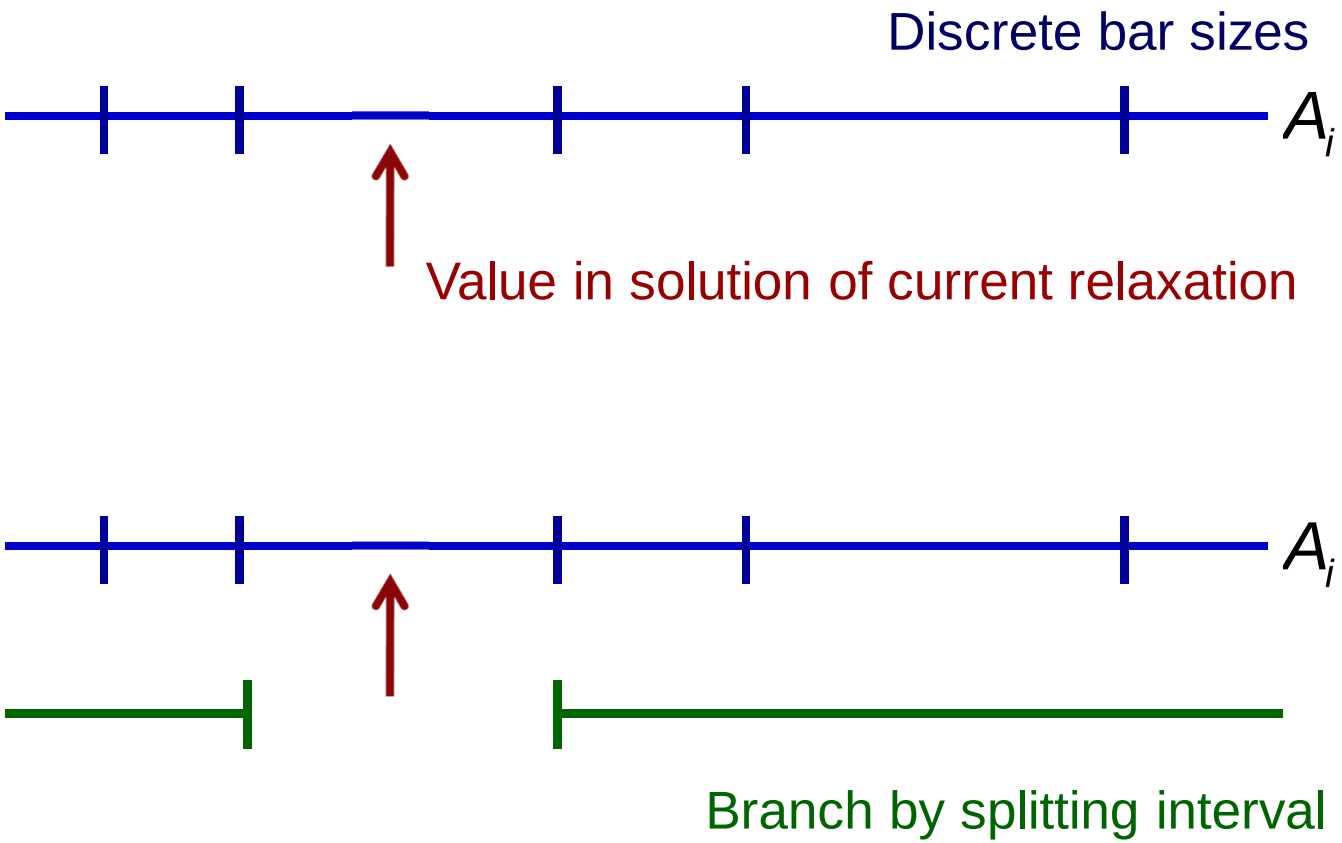
- Use the **original model** (don't introduce new variables)
- Branch by **splitting** the range of areas A_i
    - No 0-1 or integer variables!
- Generate a linear **quasi-relaxation**, which is much smaller than MILP model.
- Use **logic cuts**.

Original hand-coded method: Bollapragada, Ghattas, and JNH 2001.

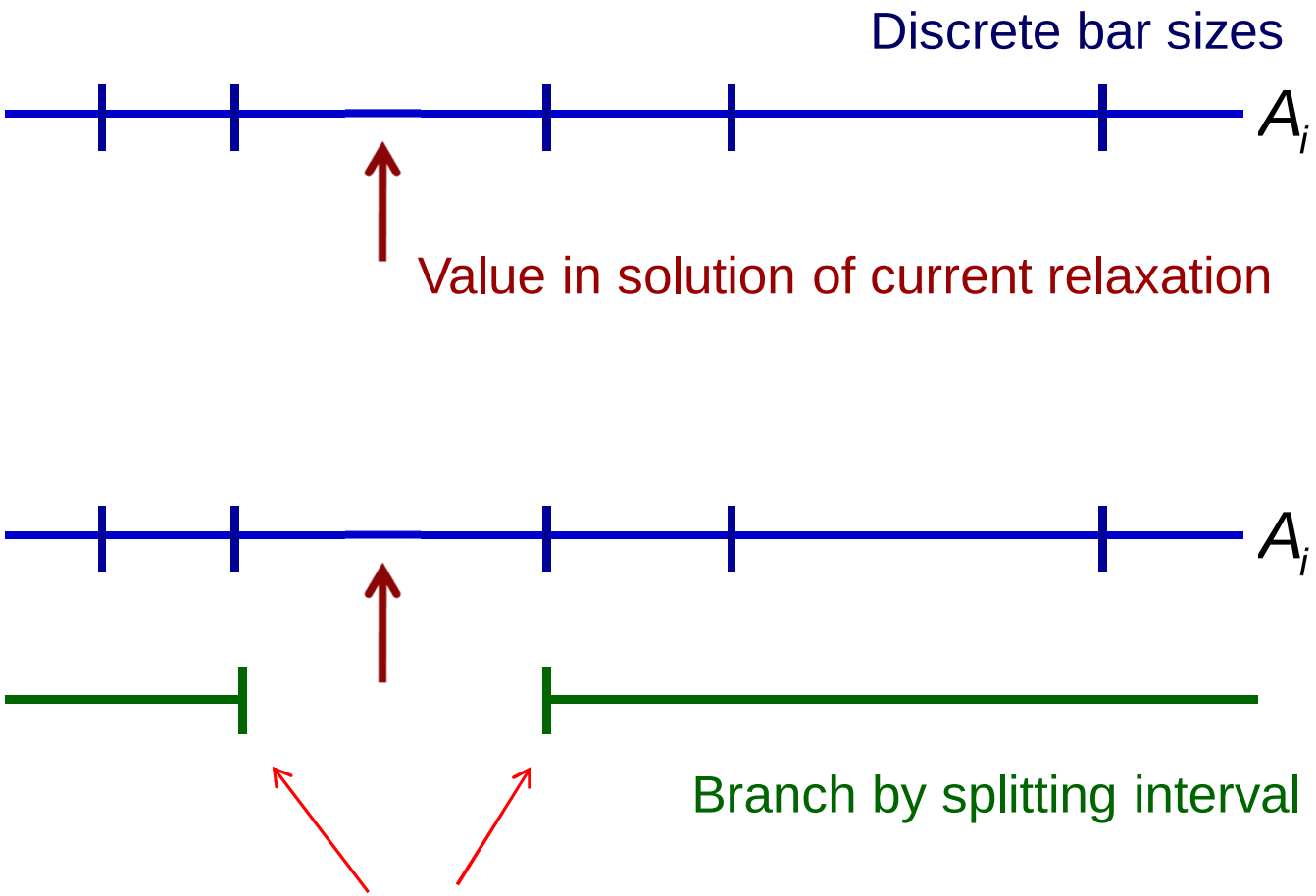
Branching


Discrete bar sizes
A_i
Value in solution of current relaxation

Branching


Discrete bar sizes
A_i
Value in solution of current relaxation
A_i
Branch by splitting interval

Branching



Solution of next relaxation likely to be at an endpoint.
This branching intelligence unavailable in 0-1 model.

Quasi-relaxation

Given problem $\min_{x \in S} \{ f(x) \}$

The problem $\min_{x' \in S'} \{ f'(x') \}$ is a **quasi-relaxation** if

for any $x \in S$, there is an $x' \in S'$ with $f'(x') \leq f(x)$.

A quasi-relaxation need not be a valid relaxation.

But its **optimal value** is a **valid lower bound** on the optimal value of the original problem.

Quasi-relaxation

Consider the problem

$$\min f(x)$$
$$g^j(x, y_j) \leq 0, \text{ all } j$$
$$x \in \mathbb{R}^n, y_j \text{ discrete}$$

Quasi-relaxation

Consider the problem

$$\min f(x)$$

$$g^j(x, y_j) \le 0, \text{ all } j$$

$$x \in \mathbb{R}^n, y_j \text{ discrete}$$

Each g^j is a vector of functions

Quasi-relaxation

Consider the problem

$$\min f(x)$$
$$g^j(x, y_j) \le 0, \text{ all } j$$
$$x \in \mathbb{R}^n, y_j \text{ discrete}$$

Each g^j is a vector of functions

Each y_j is a scalar variable

Quasi-relaxation

Consider the problem $\min f(x)$

$$g^j(x, y_j) \le 0, \text{ all } j$$

$$x \in \mathbb{R}^n, y_j \text{ discrete}$$

Each g^j is a vector of functions

Each y_j is a scalar variable

Relaxing the problem by making y_j continuous could result in a **nonconvex** problem.

Quasi-relaxation

Consider the problem $\min f(x)$

$$g^j(x, y_j) \le 0, \text{ all } j$$

$$x \in \mathbb{R}^n, \; y_j \text{ discrete}$$

Each g^j is a vector of functions

Each y_j is a scalar variable

Relaxing the problem by making y_j continuous could result in a **nonconvex** problem.

But suppose the problem becomes convex when each y_j is fixed to a **constant**.

Quasi-relaxation

Consider the problem $\min f(x)$

$$g^j(x, y_j) \le 0, \text{ all } j$$

$$x \in \mathbb{R}^n, \; y_j \text{ discrete}$$

Each g^j is a vector of functions

Each y_j is a scalar variable

Relaxing the problem by making y_j continuous could result in a **nonconvex** problem.

But suppose the problem becomes convex when each y_j is fixed to a **constant**.

Then we may be able to write a **convex quasi-relaxation**.

Quasi-relaxation

Consider the problem

$$\min f(x)$$
$$g^j(x, y_j) \le 0, \text{ all } j$$
$$x \in \mathbb{R}^n, y_j \text{ discrete}$$

Theorem (JNH 2005)

If each $g^j(x,y)$ is semihomogeneous in x and concave in scalar y_j, then the following is a **quasi-relaxation**:

$$\min f(x)$$
$$g^j(x^1, y_j^L) + g^j(x^2, y_j^U) \le 0, \text{ all } j$$
$$\alpha x^L \le x^1 \le \alpha x^U$$
$$(1-\alpha)x^L \le x^2 \le (1-\alpha)x^U$$
$$x = x^1 + x^2$$
$$\alpha \in [0,1]$$

Quasi-relaxation

Consider the problem

$$\min f(x)$$
$$g^j(x, y_j) \le 0, \text{ all } j$$
$$x \in \mathbb{R}^n, y_j \text{ discrete}$$

Theorem (JNH 2005)

If each $g^j(x,y)$ is semihomogeneous in x and concave in scalar y_j, then the following is a **quasi-relaxation**:

$$\min f(x)$$
$$g^j(x^1, y_j^L) + g^j(x^2, y_j^U) \le 0, \text{ all } j$$
$$\alpha x^L \le x^1 \le \alpha x^U$$
$$(1-\alpha)x^L \le x^2 \le (1-\alpha)x^U$$
$$x = x^1 + x^2$$
$$\alpha \in [0,1]$$

$g(\alpha x, y) \le \alpha g(x, y)$ for all x, y and $\alpha \in [0,1]$

$g(0, y) = 0$ for all y

Quasi-relaxation

Consider the problem

$$\min f(x)$$
$$g^j(x, y_j) \le 0, \text{ all } j$$
$$x \in \mathbb{R}^n, y_j \text{ discrete}$$

Theorem (JNH 2005)

If each $g^j(x,y)$ is semihomogeneous in x and concave in scalar y_j, then the following is a **quasi-relaxation**:

$$\min f(x)$$
$$g^j(x^1, \boxed{y_j^L}) + g^j(x^2, \boxed{y_j^U}) \le 0, \text{ all } j$$
$$\alpha x^L \le x^1 \le \alpha x^U$$
$$(1-\alpha)x^L \le x^2 \le (1-\alpha)x^U$$
$$x = x^1 + x^2$$
$$\alpha \in [0,1]$$

Bounds on y_i

Quasi-relaxation

Consider the problem

$$\min f(x)$$
$$g^j(x, y_j) \le 0, \text{ all } j$$
$$x \in \mathbb{R}^n, y_j \text{ discrete}$$

Theorem (JNH 2005)

If each $g^j(x,y)$ is semihomogeneous in x and concave in scalar y_j, then the following is a **quasi-relaxation**:

$$\min f(x)$$
$$g^j(x^1, y_j^L) + g^j(x^2, y_j^U) \le 0, \text{ all } j$$
$$\alpha x^L \le x^1 \le \alpha x^U$$
$$(1-\alpha)x^L \le x^2 \le (1-\alpha)x^U$$
$$x = x^1 + x^2$$
$$\alpha \in [0,1]$$

Bounds on x

Quasi-relaxation

Why?

Take any feasible solution $\bar{x}, \bar{y}$

$$\min f(x)$$

$$g^j(x^1, y_j^L) + g^j(x^2, y_j^U) \le 0, \text{ all } j$$

$$\alpha x^L \le x^1 \le \alpha x^U$$

$$(1-\alpha)x^L \le x^2 \le (1-\alpha)x^U$$

$$x = x^1 + x^2$$

$$\alpha \in [0,1]$$

Quasi-relaxation

Why?

Take any feasible solution $\bar{x}, \bar{y}$

Choose α so that

$$\bar{y}_j = \alpha_j y_j^L + (1-\alpha_j) y_j^U$$

Set $x^{j1} = \alpha_j \bar{x}, \quad x^{j2} = (1-\alpha_j)\bar{x}$

$$\min f(x)$$

$$g^j(x^1, y_j^L) + g^j(x^2, y_j^U) \le 0, \text{ all } j$$

$$\alpha x^L \le x^1 \le \alpha x^U$$

$$(1-\alpha)x^L \le x^2 \le (1-\alpha)x^U$$

$$x = x^1 + x^2$$

$$\alpha \in [0,1]$$

Quasi-relaxation

$$\min f(x)$$
$$g^j(x^1, y_j^L) + g^j(x^2, y_j^U) \le 0, \quad \text{all } j$$
$$\alpha x^L \le x^1 \le \alpha x^U$$
$$(1-\alpha)x^L \le x^2 \le (1-\alpha)x^U$$
$$x = x^1 + x^2$$
$$\alpha \in [0,1]$$

Why?

Take any feasible solution $\bar{x}, \bar{y}$

Choose α so that

$$\bar{y}_j = \alpha_j y_j^L + (1-\alpha_j) y_j^U$$

Set $x^{j1} = \alpha_j \bar{x}, \quad x^{j2} = (1-\alpha_j)\bar{x}$

Then for each component *i* of g^j we have

$$g_i^j(x^{j1}, y_j^L) + g_i^j(x^{j2}, y_j^U) = g_i^j\left(\alpha_j \bar{x}, y_j^L\right) + g_i^j\left((1-\alpha_j)\bar{x}, y_j^U\right)$$
$$= \alpha_j g_i^j(x, y_j^L) + (1-\alpha_j) g_i^j(x, y_j^U) \le g_i^j\left(x, \alpha_j y_j^L + (1-\alpha_j) y_j^U\right) = g_i^j(x, y_j)$$

homogeneity

concavity

Quasi-relaxation

$$\min f(x)$$
$$g^j(x, y_j) \le 0, \text{ all } j$$
$$x \in \mathbb{R}^n, y_j \text{ discrete}$$

So we have a feasible solution of the quasi-relaxation with value that is less than or equal to (in fact equal to) that of the original problem.

satisfied, by above argument

$$\min f(x)$$
$$g^j(x^1, y_j^L) + g^j(x^2, y_j^U) \le 0, \text{ all } j$$
$$\alpha x^L \le x^1 \le \alpha x^U$$
$$(1-\alpha)x^L \le x^2 \le (1-\alpha)x^U$$
$$x = x^1 + x^2$$
$$\alpha \in [0,1]$$

satisfied, by construction

Quasi-relaxation

$$\min f(x)$$
$$g^j(x, y_j) \leq 0, \text{ all } j$$
$$x \in \mathbb{R}^n, y_j \text{ discrete}$$

$\frac{E_i}{h_i} A_i v_i = s_i$ has the form $g(x,y) = 0$ with g semihomogenous in x and concave (linear) in y because we can write it

$$\frac{E_i}{h_i} A_i v_i - s_i = 0$$

with $x = (v_i, s_i)$, $y = A_i$.

Truss Structure Design

So we have a quasi-relaxation of the truss problem:

$$\min \quad \sum_i h_i [A_i^L y_i + A_i^U (1 - y_i)]$$

$$s.t. \quad \sum_i \cos\theta_{ij} s_i = p_j, \text{ all } j$$

$$\sum_j \cos\theta_{ij} d_j = v_{i0} + v_{i1}, \text{ all } i$$

$$\frac{E_i}{h_i}(A_i^L v_{i0} + A_i^U v_{i1}) = s_i, \text{ all } i$$

$$v_i^L y_i \le v_{i0} \le v_i^U y_i, \text{ all } i$$

$$v_i^L (1 - y_i) \le v_{i1} \le v_i^U (1 - y_i), \text{ all } i$$

$$d_j^L \le d_j \le d_j^U, \text{ all } j$$

$$0 \le y_i \le 1, \text{ all } i$$

Hooke's law is linearized

Elongation bounds split into 2 sets of bounds

Truss Structure Design

Logic cuts

v_{i0} and v_{i1} must have same sign in a feasible solution.

If not, we branch by adding logic cuts

$$v_{i0}, v_{i1} \leq 0, \qquad v_{i0}, v_{i1} \geq 0$$

Truss Structure Design

In general, we can have a **metaconstraint** to represent the semihomogeneous constraint $g(x,y) \leq 0$.

This tells the solver to generate a quasi-relaxation.

Truss Structure Design

In general, we can have a **metaconstraint** to represent the semihomogeneous constraint $g(x,y) \leq 0$.

This tells the solver to generate a quasi-relaxation.

Since a bilinear constraint $xy = \alpha$ is always semihomogeneous, we will use a **bilinear** metaconstraint with a quasi-relaxation option.

Truss Structure Design

SIMPL model

Recognized as linear systems

```
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
12.     relaxation = { lp:quasi } }
13. SEARCH
14.   type = { bb:bestdive }
15.   branching = { hooke:first:quasicut, A:splitup }
```

Truss Structure Design

SIMPL model

```
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
12.     relaxation = { lp:quasi } }
13. SEARCH
14.   type = { bb:bestdive }
15.   branching = { hooke:first:quasicut, A:splitup }
```

Recognized as bilinear system

Truss Structure Design

SIMPL model

```
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
12.     relaxation = { lp:quasi } }
13. SEARCH
14.   type = { bb:bestdive }
15.   branching = { hooke:first:quasicut, A:splitup }
```

Generate quasi-relaxation for semihomogenous function

Truss Structure Design

SIMPL model

```
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
12.     relaxation = { lp:quasi } }
13. SEARCH
14.   type = { bb:bestdive }
15.   branching = { hooke:first:quasicut, A:splitup }
```

Branch first on violated logic cuts for quasi-relaxation

Truss Structure Design

SIMPL model

```
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
12.     relaxation = { lp:quasi } }
13. SEARCH
14.   type = { bb:bestdive }
15.   branching = { hooke:first:quasicut, A:splitup }
```

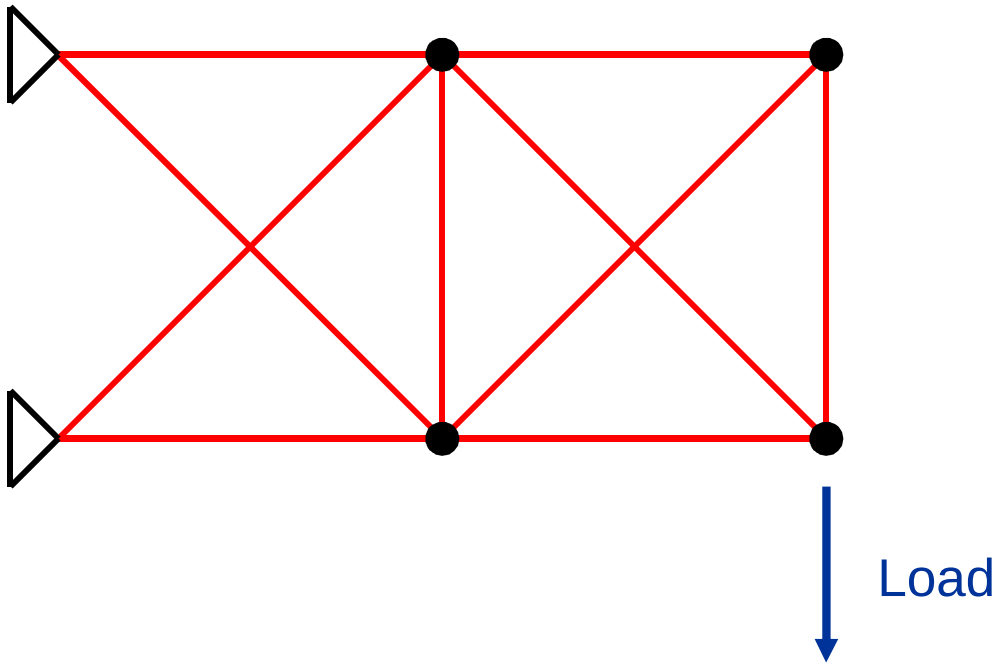
Then branch on A_i in-domain constraint.

Violated when A_i is not one of the discrete bar sizes.

Take upper branch first.

Truss Structure Design

10-bar cantilever truss



Truss Structure Design

Hand-coded integrated method

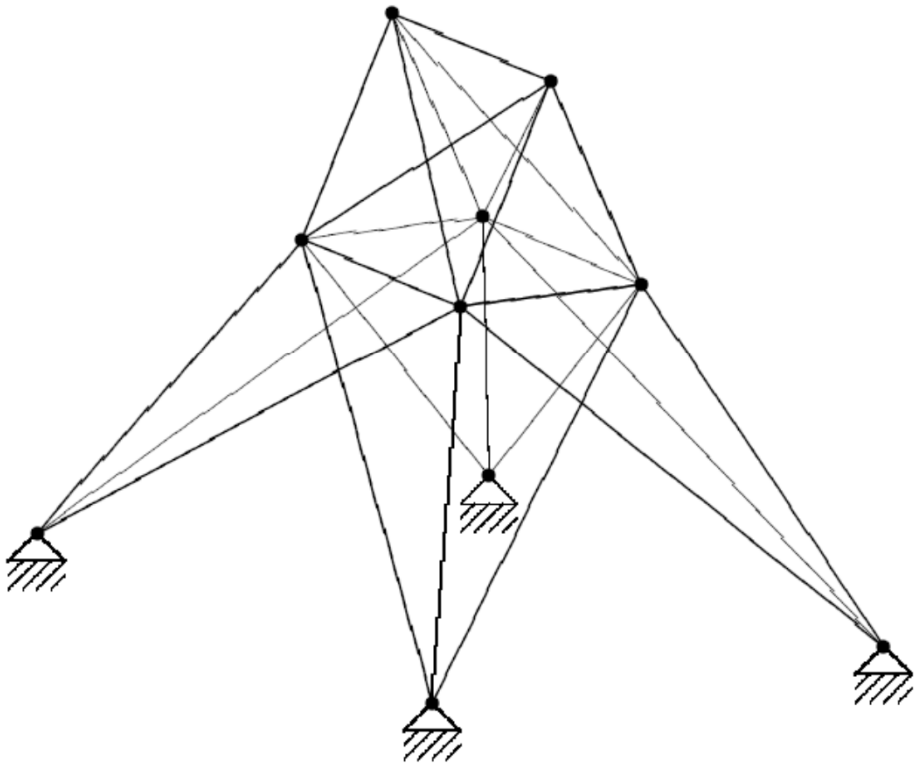
Computational results (seconds)

No. bars	Loads	BARON	CPLEX 11	Hand coded	SIMPL
10	1	5.3	0.40	0.03	0.08
10	1	3.8	0.26	0.02	0.07
10	1	8.1	0.83	0.16	0.49
10	1	8.8	1.2	0.22	0.63
10	2	24	4.9	0.64	1.84
10	2*	327	146	145	65
10	2*	2067	1087	600	651

*plus displacement bounds

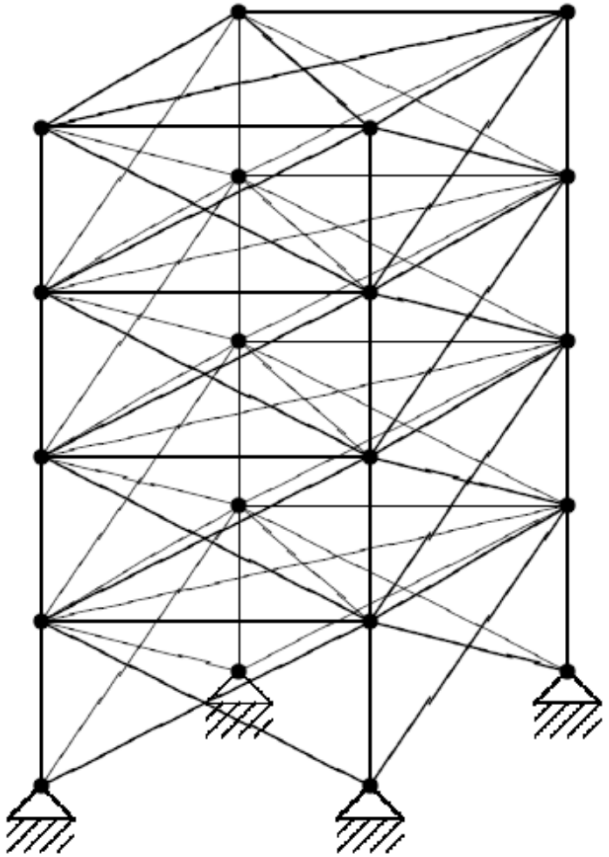
Truss Structure Design

25-bar problem



Truss Structure Design

72-bar problem



Truss Structure Design

Hand-coded integrated method

Computational results (seconds)

No. bars	Loads	BARON	CPLEX 11	Hand coded	SIMPL
25	2	3,302	44	44	20
72	2	3,376	208	33	28
90	2	21,011	570	131	92
108	2	> 24 hr*	3208	1907	1720
200	2	> 24 hr*	> 24 hr*	> 24 hr**	> 24 hr***

* no feasible solution found

** best feasible solution has cost 32,748

*** best feasible solution has cost 32,700

Current Version of SIMPL

- To download:
  - Click the link to SIMPL on John Hooker's website.
  - See readme file for complete instructions.
  - Download executable and associated files
- Operational on GNU/Linux only
- Requires subsidiary solvers
  - CPLEX
  - Eclipse (free download)
- Download problem instances
  - Including all reported in this talk.