

Constraint Programming Techniques in MINLP

J. N. Hooker

MINLP Workshop
Pittsburgh, June 2014

What can CP contribute to MINLP?

- **Interval propagation** (range reduction).
 - Already used in global optimization solvers.
- Focus on lesser-known techniques:
 - **Domain filtering** with Lagrange multipliers.
 - Efficient representation of **piecewise linear** functions.
 - Branching on **multiple discrete values**.
 - Bounds from **quasi-relaxations**.
 - Optimization/propagation in **decision diagrams**.
 - Management of **McCormick factors** with global constraints and semantic typing (*no time for this today*).

Constraint Programming Perspective

- All (successful) optimization method combine **search** with **relaxation** and **inference**.
- Math programming focuses on **relaxation**.
 - LP, Lagrangean, etc.
- Constraint programming (CP) focuses on **inference**.
 - Domain filtering, constraint propagation

Constraint Programming Perspective

- All (successful) optimization method combine **search** with **relaxation** and **inference**.
- Math programming focuses on **relaxation**.
 - LP, Lagrangean, etc.
- Constraint programming (CP) focuses on **inference**.
 - Domain filtering, constraint propagation
- Math programming uses inference...
 - To generate cutting planes, Benders cuts, etc.
 - But the purpose is to strengthen the **relaxation**.

Constraint Programming Perspective

- CP uses inference for **consistency maintenance**
 - ...rather than to strengthen a relaxation.
- Greater consistency means **less backtracking** during search.
 - The concept of consistency never developed in math programming, but it helps to explain search behavior.

Constraint Programming Perspective

- CP uses inference for **consistency maintenance**
 - ...rather than to strengthen a relaxation.
- Greater consistency means **less backtracking** during search.
 - The concept of consistency never developed in math programming, but it helps to explain search behavior.
- Several types of consistency
 - Domain consistency (generalized arc consistency)
 - Bounds consistency
 - Strong k-consistency
 - Etc. etc.

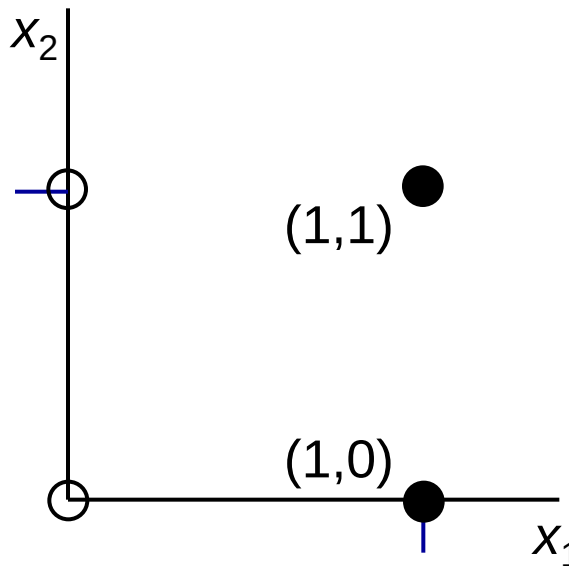
Domain consistency

A constraint set is **domain consistent** if the domain of each variable x_i is the projection of the feasible set onto x_i .

$$x_1 + x_2 \geq 1$$

$$x_1 - x_2 \geq 0$$

$$x_1, x_2 \in \{0,1\}$$



Domain consistency

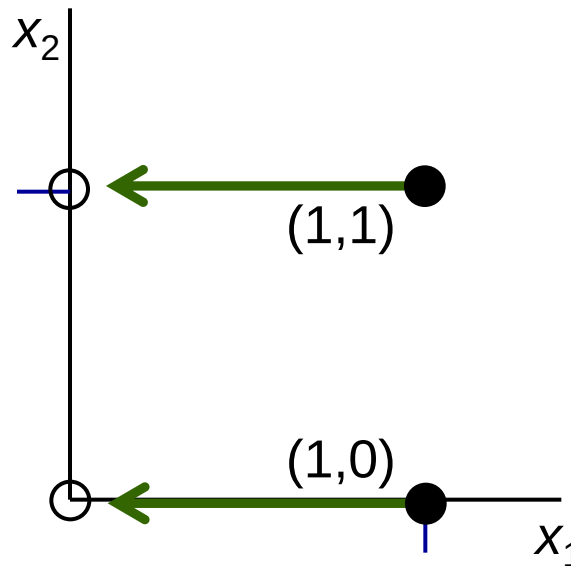
A constraint set is **domain consistent** if the domain of each variable x_i is the projection of the feasible set onto x_i .

$$x_1 + x_2 \geq 1$$

$$x_1 - x_2 \geq 0$$

$$x_1, x_2 \in \{0,1\}$$

Projection onto $x_2 = \{0,1\}$



Domain consistency

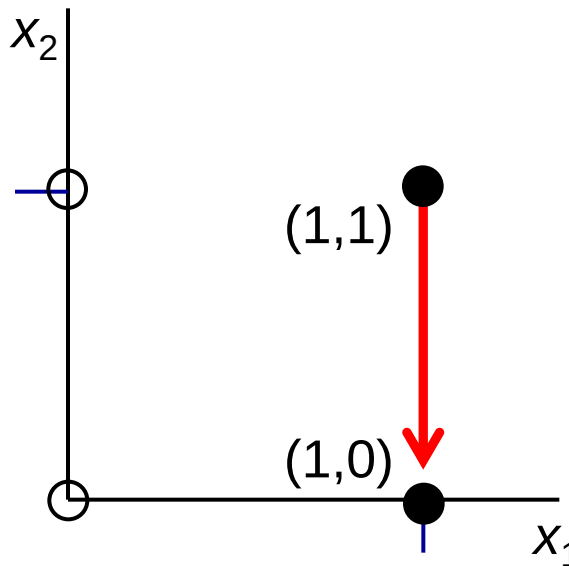
A constraint set is **domain consistent** if the domain of each variable x_i is the projection of the feasible set onto x_i .

$$x_1 + x_2 \geq 1$$

$$x_1 - x_2 \geq 0$$

$$x_1, x_2 \in \{0,1\}$$

Projection onto $x_2 = \{0,1\}$



Not domain consistent.

Projection onto $x_1 = \{1\}$

Domain consistency

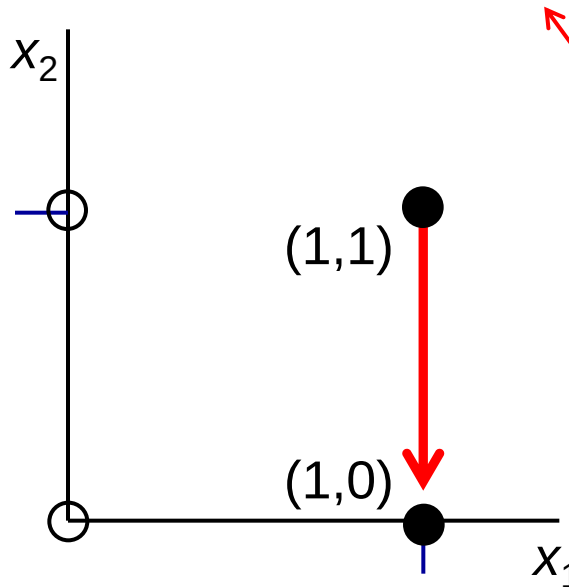
A constraint set is **domain consistent** if the domain of each variable x_i is the projection of the feasible set onto x_i .

$$x_1 + x_2 \geq 1$$

$$x_1 - x_2 \geq 0$$

$$x_1 \in \{1\}, x_2 \in \{0,1\}$$

Projection onto $x_2 = \{0,1\}$



Achieve domain consistency by **filtering** 0 from domain of x_1 .

Projection onto $x_1 = \{1\}$

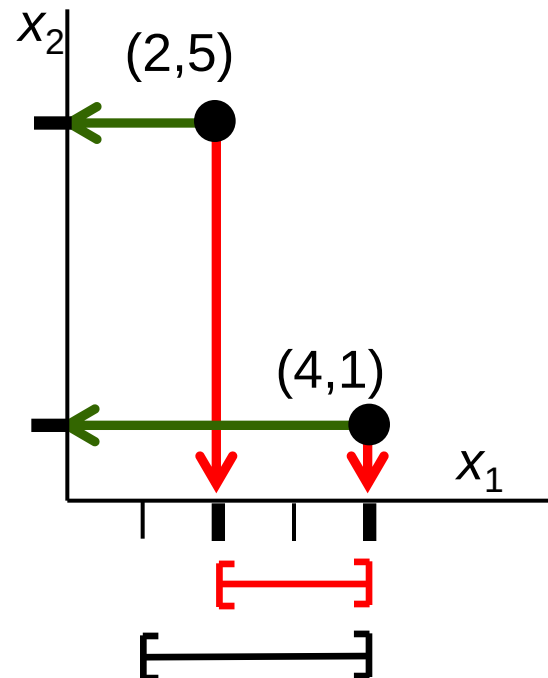
Bounds consistency

- A constraint set is **bounds consistent** if the domain of each variable x_i spans the **same interval** as the projection of the feasible set onto x_i .

$$2x_1 + x_2 = 9$$

$$x_1 \in \{1, 2, 3, 4\}$$

$$x_2 \in \{1, 5\}$$



Not bounds consistent.

Consistency maintenance

- Domain or bounds consistency is normally achieved (if at all) for **one constraint at a time**.
 - This can be NP-hard.
 - But allows one to exploit **special structure** of constraints.
 - Much as cutting planes exploit structure of certain classes of inequalities.

Consistency maintenance

- Domain or bounds consistency is normally achieved (if at all) for **one constraint at a time**.
 - This can be NP-hard.
 - But allows one to exploit **special structure** of constraints.
 - Much as cutting planes exploit structure of certain classes of inequalities.
- Particularly effective when the model consists of **global constraints**.
 - ...which represent a set of simpler constraints.
 - Alldiff, cardinality, element, nvalues, sequence, circuit, path, regular, cumulative, stretch, etc.

Propagation

- Reduced domains are passed (**propagated**) to the next constraint.
- ...which may allow further reduction.
- Generally does not achieve consistency for entire constraint set.
- But it drastically reduces backtracking.

Bounds propagation

- Bounds obtained by achieving bound consistency can be propagated.
 - This is important in global optimization (**range reduction**).

x_1

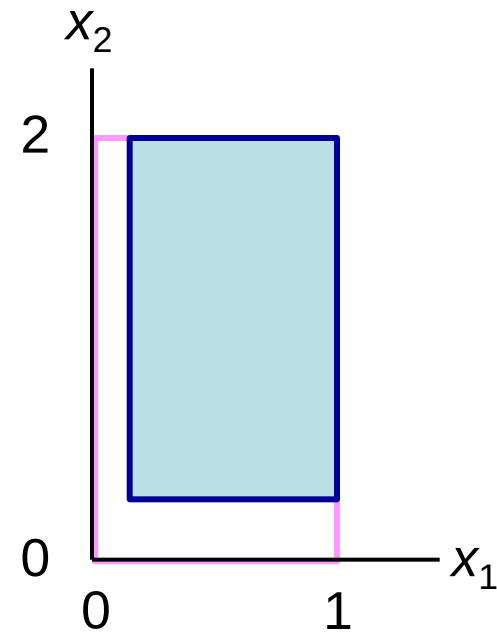
Bounds propagation

- Bounds obtained by achieving bound consistency can be propagated.
 - This is important in global optimization (**range reduction**).

- Example:
$$4x_1x_2 = 1$$
$$2x_1 + x_2 \leq 2$$
$$x_1 \in [0.125, 1]$$
$$x_2 \in [0.25, 2]$$

Filter using constraint 1: $x_1 = \frac{1}{4x_2} \geq \frac{1}{4 \cdot 2} = 0.125$

$$x_2 = \frac{1}{4x_1} \geq \frac{1}{4 \cdot 1} = 0.25$$



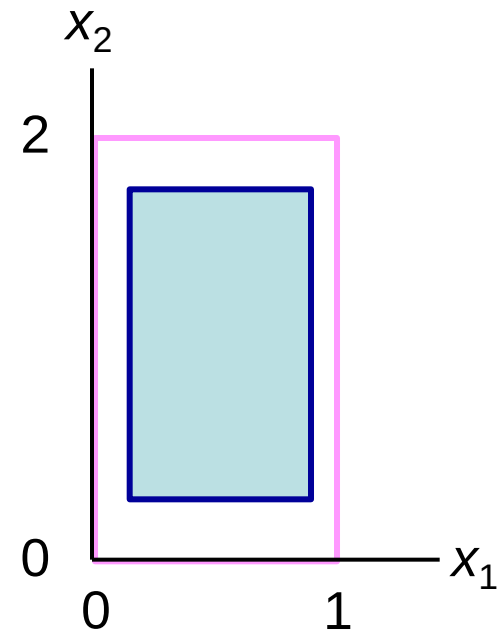
Bounds propagation

- Bounds obtained by achieving bound consistency can be propagated.
 - This is important in global optimization (**range reduction**).

- Example: $4x_1x_2 = 1$
 $2x_1 + x_2 \leq 2$
 $x_1 \in [0.125, 0.875]$
 $x_2 \in [0.25, 1.75]$

Propagate to
constraint 2:

$$x_1 \leq 1 - \frac{x_2}{2} \leq \frac{0.25}{2} = 0.875$$
$$x_2 \leq 2 - 2x_1 \leq 2 - 2 \cdot 0.125 = 1.75$$

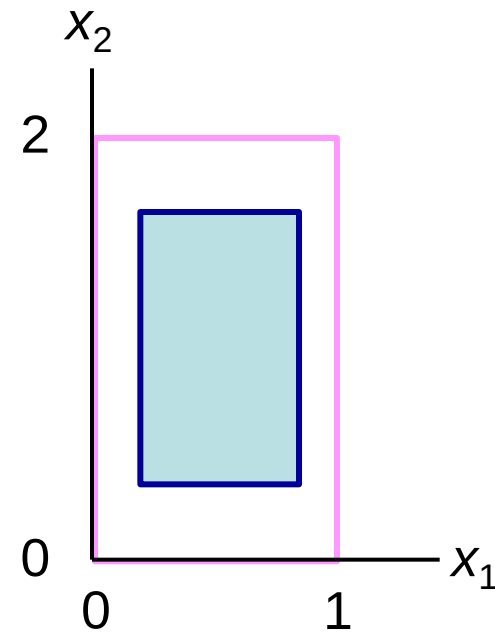


Bounds propagation

- Bounds obtained by achieving bound consistency can be propagated.
 - This is important in global optimization (**range reduction**).

- Example: $4x_1x_2 = 1$
 $2x_1 + x_2 \leq 2$
 $x_1 \in [0.146, 0.854]$
 $x_2 \in [0.293, 1.707]$

Continuing, bounds asymptotically converge:



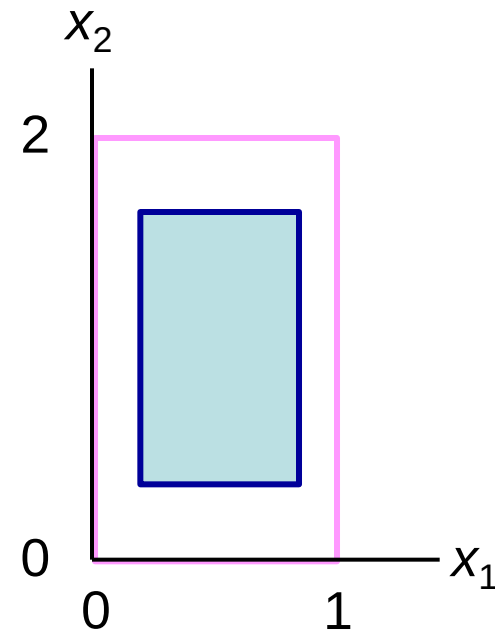
Bounds propagation

- Bounds obtained by achieving bound consistency can be propagated.
 - This is important in global optimization (**range reduction**).

- Example: $4x_1x_2 = 1$
 $2x_1 + x_2 \leq 2$
 $x_1 \in [0.146, 0.854]$
 $x_2 \in [0.293, 1.707]$

Continuing, bounds asymptotically converge:

Solvers truncate the process.



k -consistency

- k -consistency is closely related to backtracking and the **dependency graph** of a constraint set.
- A constraint set is **k -consistent** if any assignment to $k - 1$ variables that violates no constraints can be extended to an assignment to k variables without violating any constraints.

x_{j_k}

k -consistency

- Example
$$\begin{aligned}x_1 + x_2 + x_4 &\geq 1 \\x_1 - x_2 + x_3 &\geq 0 \\x_1 - x_4 &\geq 0 \\x_j &\in \{0,1\}\end{aligned}$$

- 2-consistent.

- not 3-consistent:

$(x_1, x_2) = (0, 0)$ cannot be extended to $(x_1, x_2, x_4) = (0, 0, ?)$.

Dependency graph

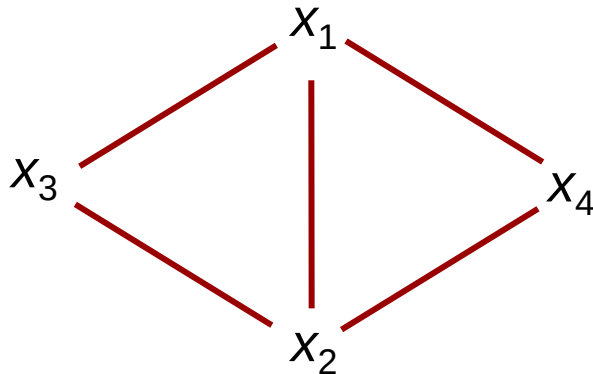
- **Dependency graph:** variables are connected by edges when they occur in a common constraint.

$$x_1 + x_2 + x_4 \geq 1$$

$$x_1 - x_2 + x_3 \geq 0$$

$$x_1 - x_4 \geq 0$$

$$x_j \in \{0,1\}$$



Dependency graph
for the example.

Dependency graph

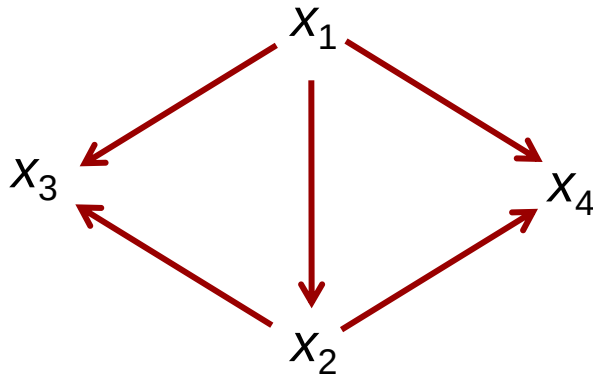
- **Dependency graph:** variables are connected by edges when they occur in a common constraint.

$$x_1 + x_2 + x_4 \geq 1$$

$$x_1 - x_2 + x_3 \geq 0$$

$$x_1 - x_4 \geq 0$$

$$x_j \in \{0,1\}$$



For a given variable ordering,
width of the graph is the
maximum in-degree

Here, width = 2
for ordering 1,2,3,4

Backtracking

- A constraint set is **strongly** k -consistent if it is i -consistent for $i = 1, \dots, k$.

Theorem (Freuder). If the dependency graph has width $< k$ for some variable ordering, then branching (in that order) solves a strongly k -consistent problem **without backtracking**.

Backtracking

- The example doesn't satisfy the conditions of the theorem.

- Width = 2, not strongly 3-consistent.

- Backtracking occurs when we set

$$(x_1, x_2, x_3, x_4) = (0, 0, 0, ?)$$

$$x_1 + x_2 + x_4 \geq 1$$

$$x_1 - x_2 + x_3 \geq 0$$

$$x_1 - x_4 \geq 0$$

$$x_j \in \{0, 1\}$$

Backtracking

- Suppose we add two constraints:
 - This is strongly 3-consistent.
 - Backtracking does not occur.

$$x_1 + x_2 + x_4 \geq 1$$

$$x_1 - x_2 + x_3 \geq 0$$

$$x_1 - x_4 \geq 0$$

$$x_1 + x_2 \geq 1$$

$$x_1 + x_3 \geq 1$$

$$x_j \in \{0,1\}$$

Backtracking

- Suppose we add two constraints:

- This is strongly 3-consistent.
- Backtracking does not occur.

- These are valid cuts!

- Cuts reduce backtracking by increasing the degree of consistency as well as by strengthening the LP relaxation.

$$x_1 + x_2 + x_4 \geq 1$$

$$x_1 - x_2 + x_3 \geq 0$$

$$x_1 - x_4 \geq 0$$

$$x_1 + x_2 \geq 1$$

$$x_1 + x_3 \geq 1$$

$$x_j \in \{0,1\}$$

Global optimization example

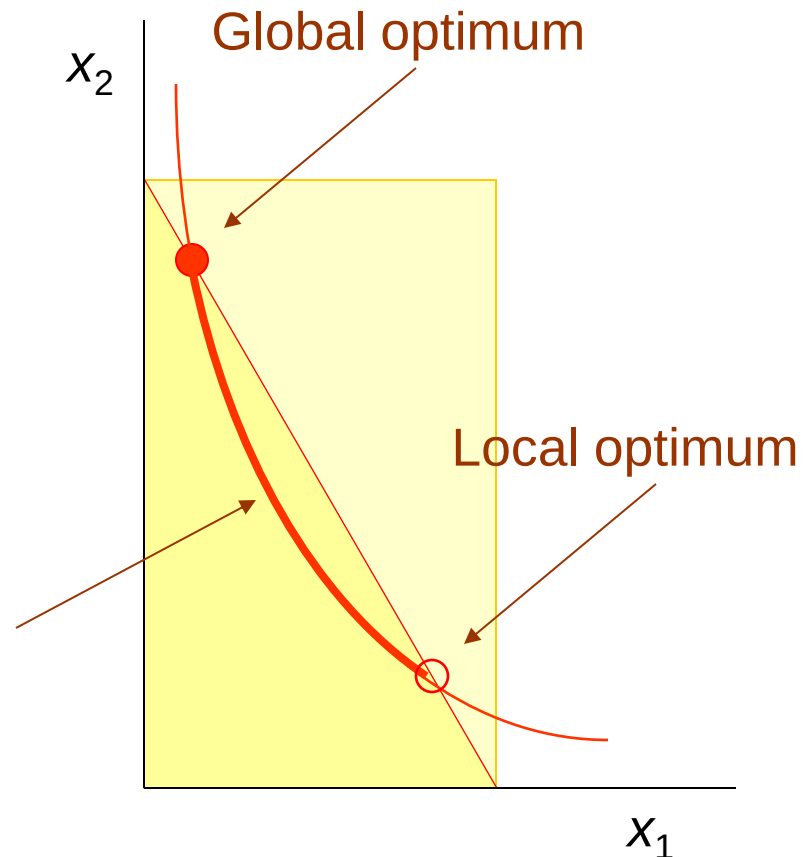
$$\max x_1 + x_2$$

$$4x_1x_2 = 1$$

$$2x_1 + x_2 \leq 2$$

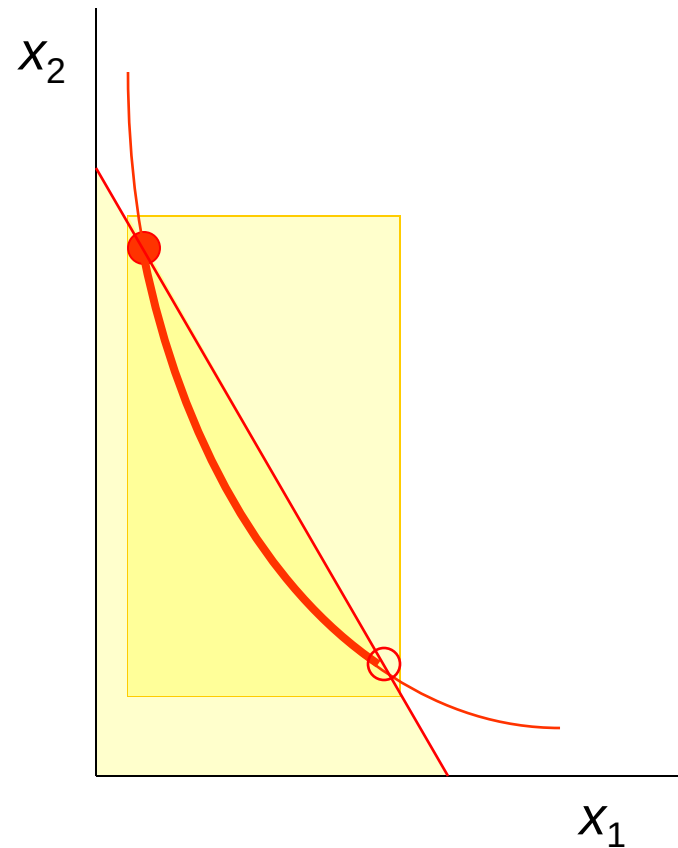
$$x_1 \in [0,1], \quad x_2 \in [0,2]$$

Feasible set



Interval propagation (range reduction)

Propagate intervals
 $[0,1]$, $[0,2]$
through constraints
to obtain
 $[1/8, 7/8]$, $[1/4, 7/4]$



Relaxation (function factorization)

Factor complex functions into elementary functions that have known linear relaxations (**McCormick factors**).

Write $4x_1x_2 = 1$ as $4y = 1$ where $y = x_1x_2$.

This factors $4x_1x_2$ into linear function $4y$ and bilinear function x_1x_2 .

Linear function $4y$ is its own linear relaxation.

Relaxation (function factorization)

Factor complex functions into elementary functions that have known linear relaxations (**McCormick factors**).

Write $4x_1x_2 = 1$ as $4y = 1$ where $y = x_1x_2$.

This factors $4x_1x_2$ into linear function $4y$ and bilinear function x_1x_2 .

Linear function $4y$ is its own linear relaxation.

Bilinear function $y = x_1x_2$ has relaxation:

$$\underline{x}_2x_1 + \underline{x}_1x_2 - \underline{x}_1\underline{x}_2 \leq y \leq \underline{x}_2x_1 + \bar{x}_1x_2 - \bar{x}_1\underline{x}_2$$

$$\bar{x}_2x_1 + \bar{x}_1x_2 - \bar{x}_1\bar{x}_2 \leq y \leq \bar{x}_2x_1 + \underline{x}_1x_2 - \underline{x}_1\bar{x}_2$$

where domain of x_j is $[\underline{x}_j, \bar{x}_j]$

Relaxation (function factorization)

The linear relaxation becomes:

$$\min x_1 + x_2$$

$$4y = 1$$

$$2x_1 + x_2 \leq 2$$

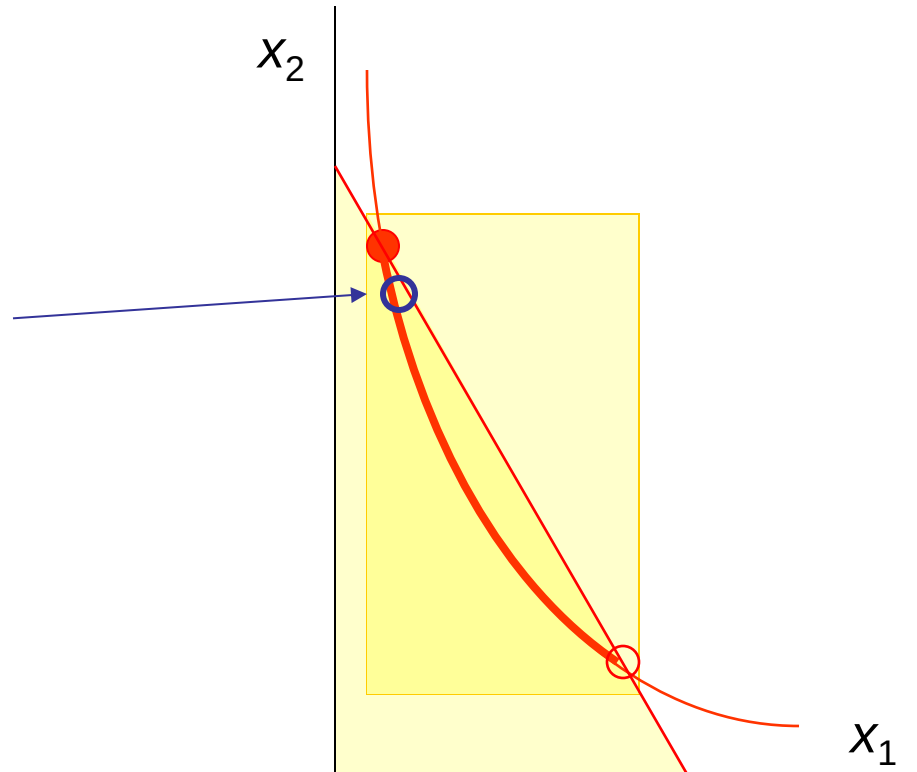
$$\underline{x}_2 x_1 + \underline{x}_1 x_2 - \underline{x}_1 \underline{x}_2 \leq y \leq \underline{x}_2 x_1 + \bar{x}_1 x_2 - \bar{x}_1 \underline{x}_2$$

$$\bar{x}_2 x_1 + \bar{x}_1 x_2 - \bar{x}_1 \bar{x}_2 \leq y \leq \bar{x}_2 x_1 + \underline{x}_1 x_2 - \underline{x}_1 \bar{x}_2$$

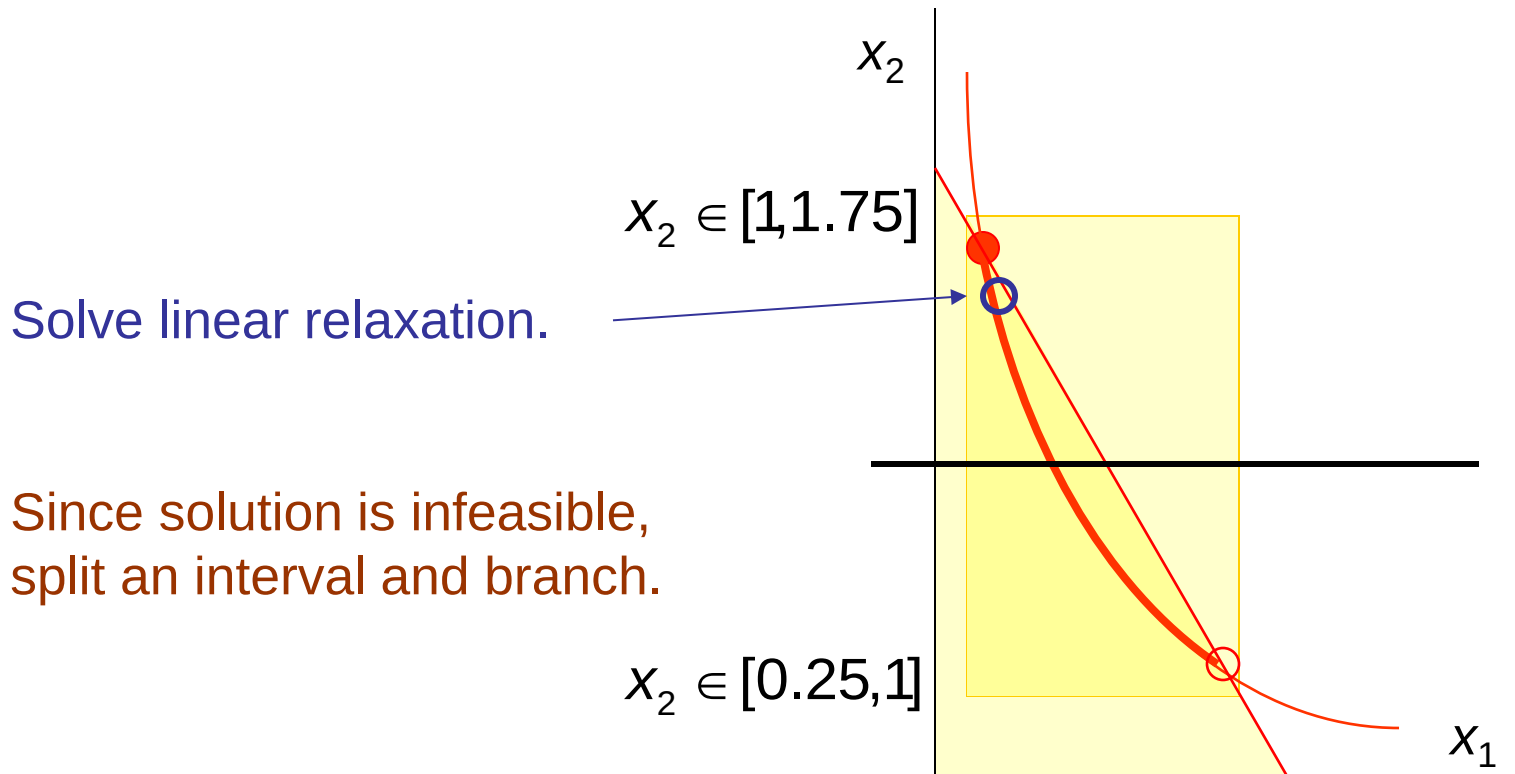
$$\underline{x}_j \leq x_j \leq \bar{x}_j, \quad j = 1, 2$$

Relaxation (function factorization)

Solve linear relaxation.

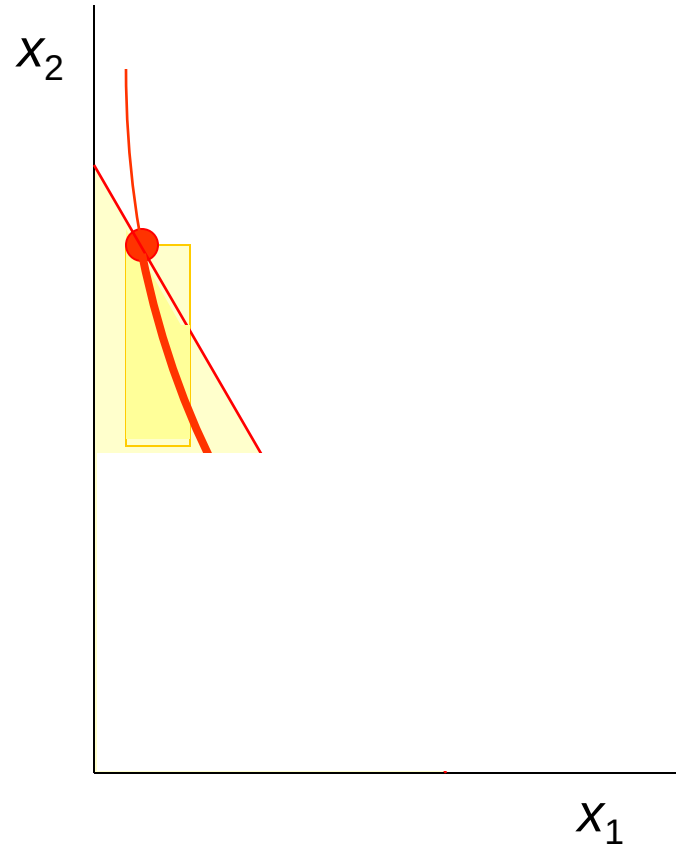
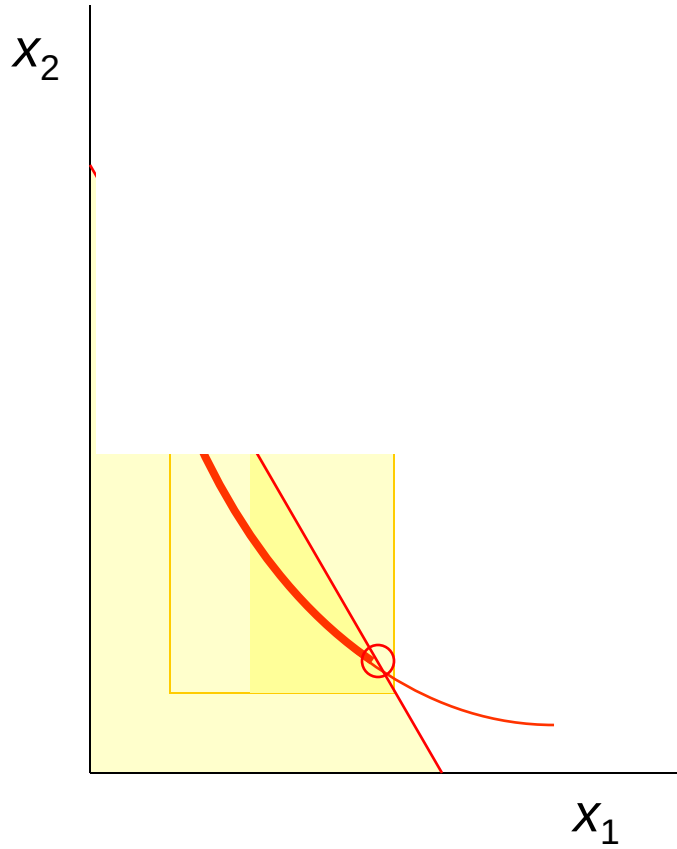


Relaxation (function factorization)



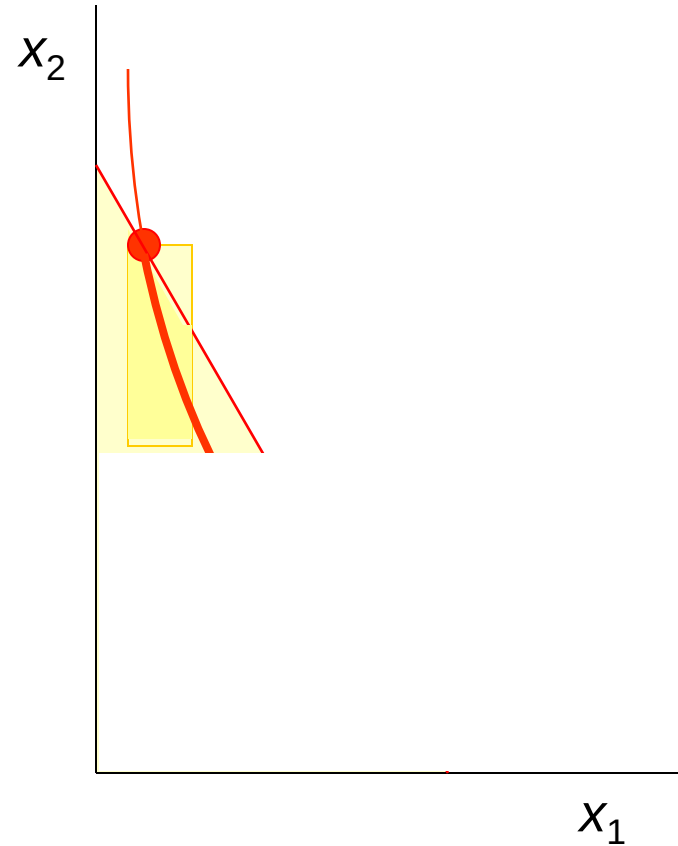
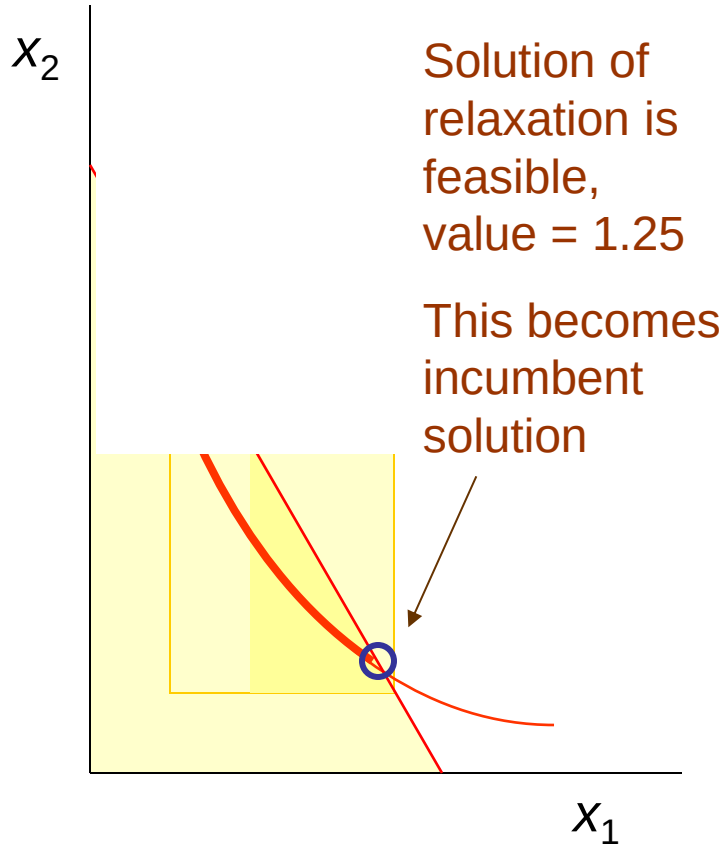
$$x_2 \in [1, 1.75]$$

$$x_2 \in [0.25, 1]$$



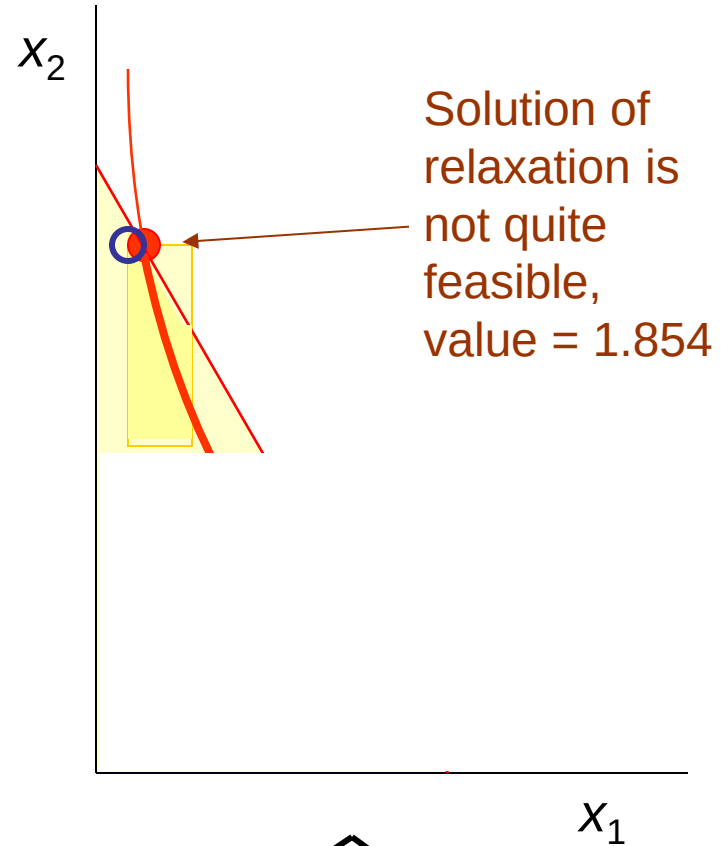
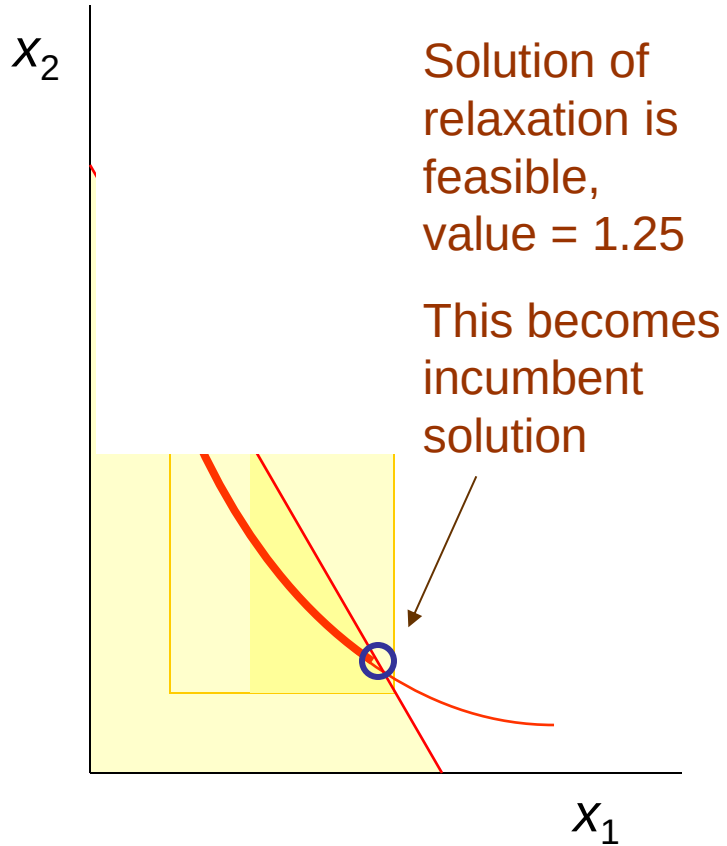
$$x_2 \in [1, 1.75]$$

$$x_2 \in [0.25, 1]$$



$$x_2 \in [1, 1.75]$$

$$x_2 \in [0.25, 1]$$



Domain filtering with Lagrange multipliers

- So far, this is all standard in global solvers.
- We can achieve stronger propagation with **filtering based on Lagrange multipliers**.
 - Reduced-cost variable fixing is a special case.

Domain filtering with Lagrange multipliers

$$\min x_1 + x_2$$

$$4y = 1$$

$$2x_1 + x_2 \leq 2$$

Associated Lagrange multiplier in solution of relaxation is $\lambda_2 = 1.1$



$$\underline{x}_2 x_1 + \underline{x}_1 x_2 - \underline{x}_1 \underline{x}_2 \leq y \leq \underline{x}_2 x_1 + \bar{x}_1 x_2 - \bar{x}_1 \underline{x}_2$$

$$\bar{x}_2 x_1 + \bar{x}_1 x_2 - \bar{x}_1 \bar{x}_2 \leq y \leq \bar{x}_2 x_1 + \underline{x}_1 x_2 - \underline{x}_1 \bar{x}_2$$

$$\underline{x}_j \leq x_j \leq \bar{x}_j, \quad j = 1, 2$$

Domain filtering with Lagrange multipliers

$$\min x_1 + x_2$$

$$4y = 1$$

$$2x_1 + x_2 \leq 2$$

Associated Lagrange multiplier in solution of relaxation is $\lambda_2 = 1.1$

$$\underline{x}_2 x_1 + \underline{x}_1 x_2 - \underline{x}_1 \underline{x}_2 \leq y \leq \underline{x}_2 x_1 + \bar{x}_1 x_2 - \bar{x}_1 \underline{x}_2$$

$$\bar{x}_2 x_1 + \bar{x}_1 x_2 - \bar{x}_1 \bar{x}_2 \leq y \leq \bar{x}_2 x_1 + \underline{x}_1 x_2 - \underline{x}_1 \bar{x}_2$$

$$\underline{x}_j \leq x_j \leq \bar{x}_j, \quad j = 1, 2$$

This yields a valid inequality for propagation:

$$2x_1 + x_2 \geq 2 - \frac{1.854 - 1.25}{1.1} = 1.451$$

Value of relaxation

Lagrange multiplier

Value of incumbent solution

Domain filtering with Lagrange multipliers

In general, suppose we have a relaxation:

$$\begin{array}{ll} \min & f(x) \\ g(x) \geq 0 & \text{with optimal solution } x^*, \text{ optimal value } v^*, \text{ and} \\ x \in S & \textbf{Lagrangian dual} \text{ solution } \lambda^*. \end{array}$$

with $\lambda_i^* > 0$, and U an upper bound on the optimal value of the original problem (perhaps from an incumbent solution).

Then we have the inequality $g_i(x) \leq \frac{U - v^*}{\lambda_i^*}$

...which can be propagated.

Domain filtering with Lagrange multipliers

A special case applies to **individual variables**:

$\min f(x)$
 $g(x) \geq 0$ has optimal solution x^* , optimal value v^* , and
 $x \in S$ **reduced gradient r .**

with $x_j^* = 0$, and U an upper bound on the optimal value of the original problem (perhaps from an incumbent solution).

Then we have the inequality $x_j \leq \frac{U - v^*}{r_j}$

...which fixes $x_j = 0$ if bound < 1 and x_j is integer
(**reduced cost variable fixing**)

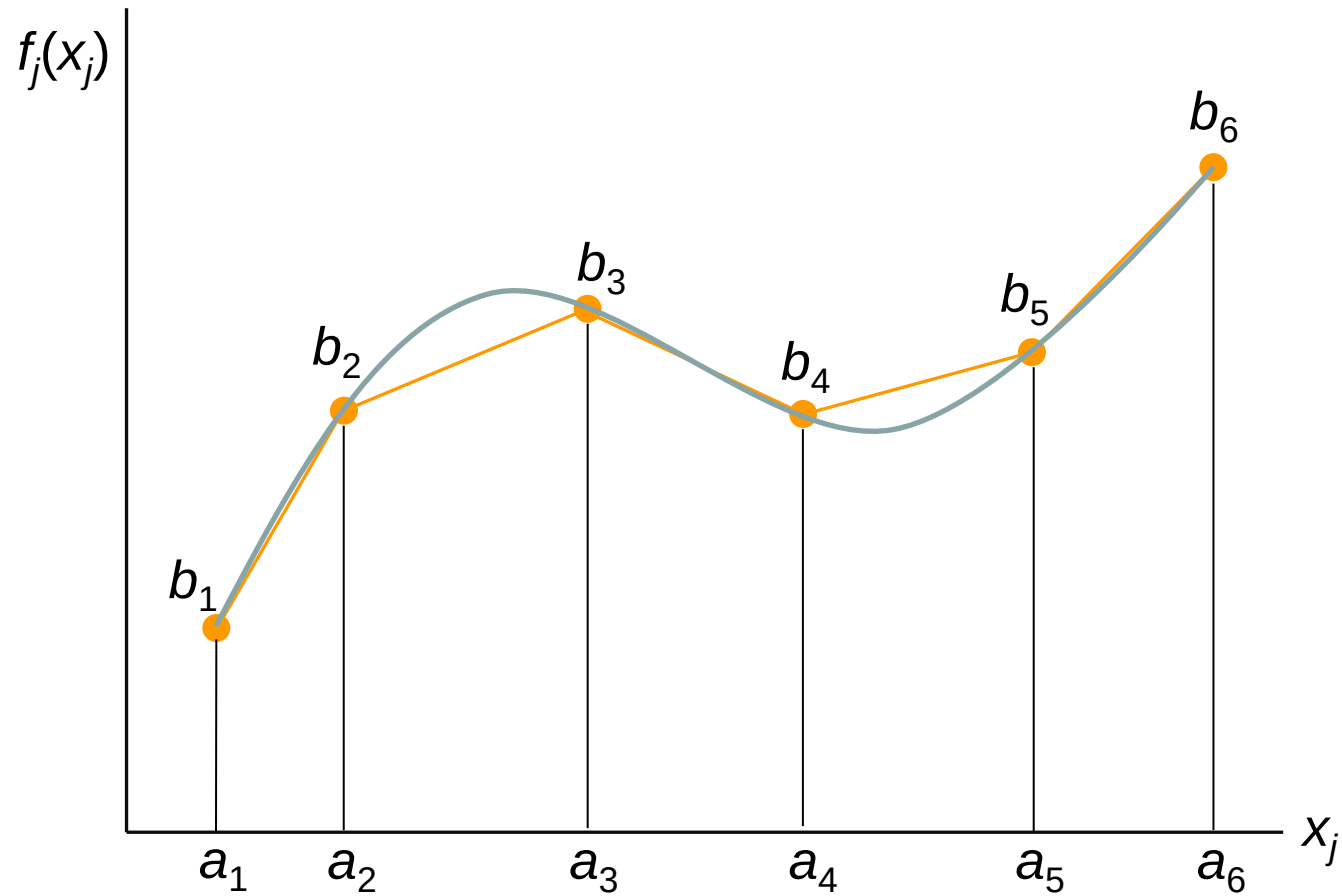
Piecewise linear functions

- Piecewise linear approximation is a powerful tool for nonlinear optimization.
 - Particularly if nonlinearities are additively separable: $f(x) = \sum_j f_j(x_j)$
- However, MINLP models require auxiliary variables.
 - A serious limitation.

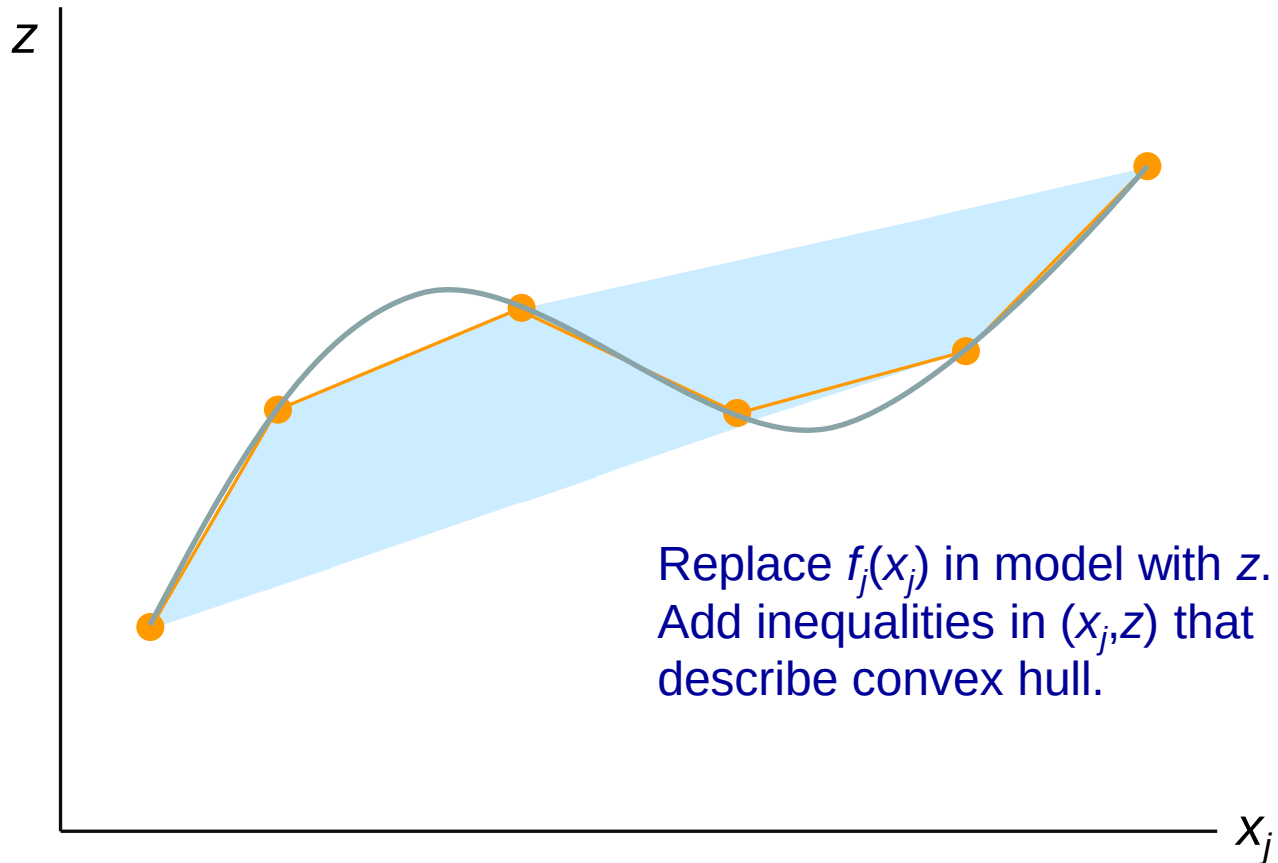
Piecewise linear functions

- CP approach **adds no variables**.
 - while providing convex hull relaxation (tight as any locally ideal MILP model)
- Use piecewise linear **global constraint** for $f_j(x_j)$:
$$\text{piecewise}(f_j, x_j, a, b)$$
 - where breakpoints are $a = (a_1, \dots, a_n)$ with $f_j(a_i) = b_i$.

Piecewise linear functions

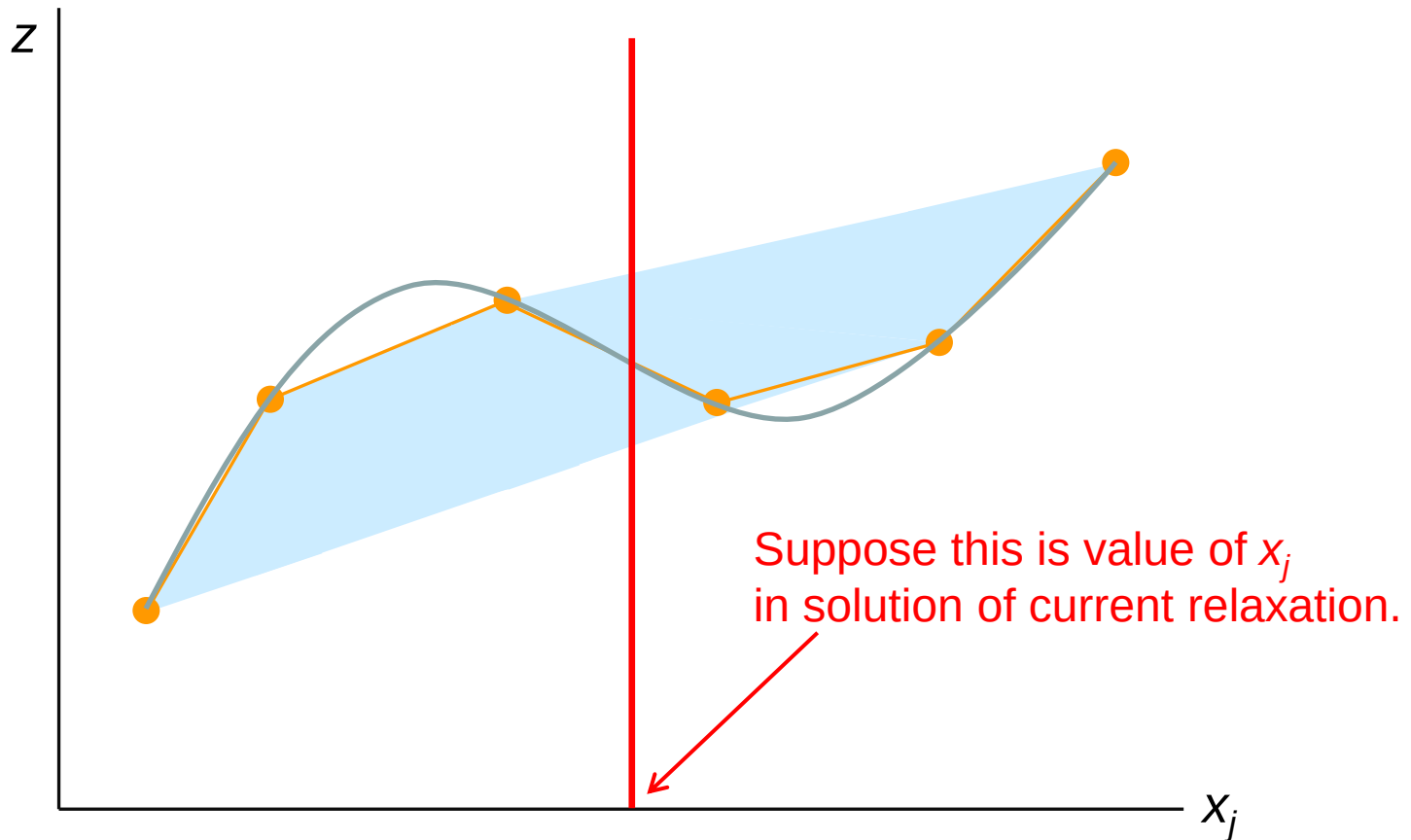


Piecewise linear functions

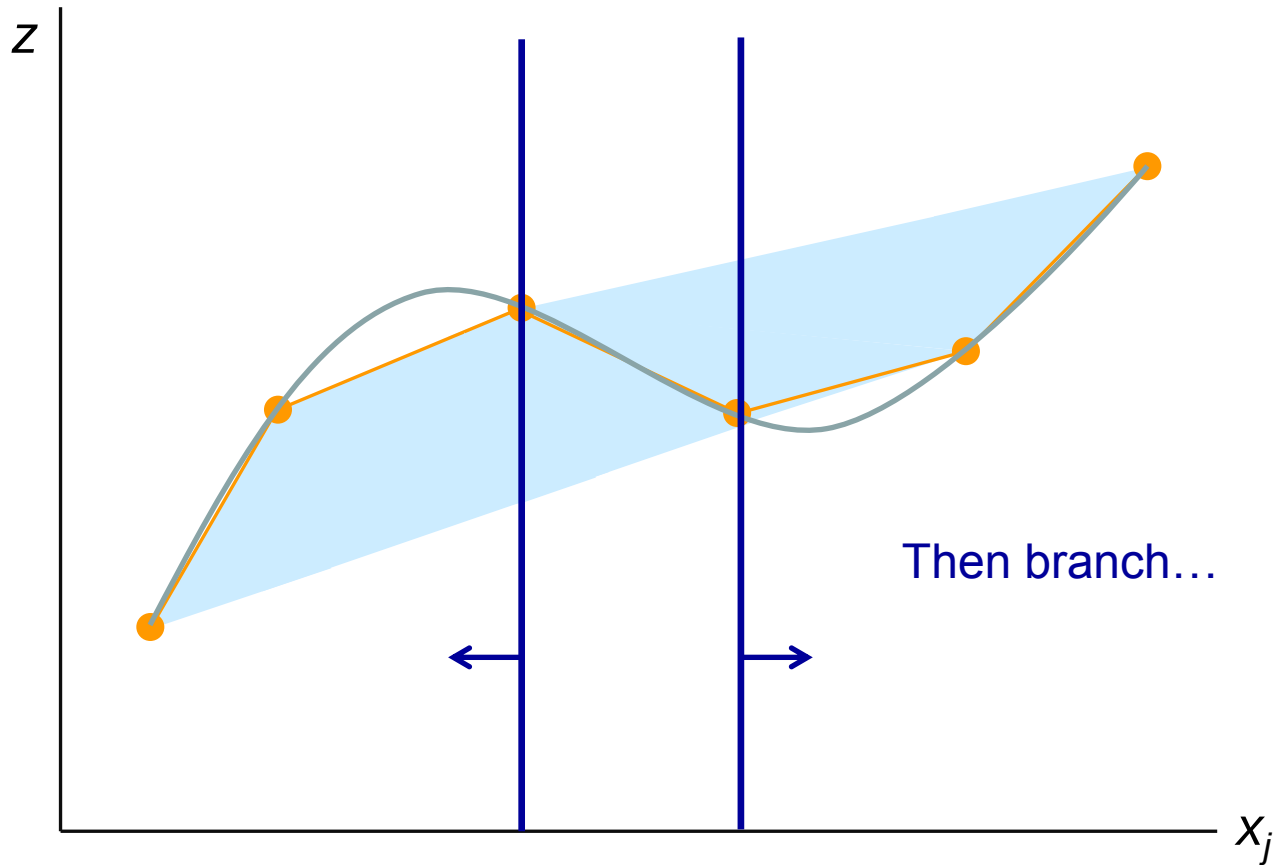


Convex hull can be computed
very rapidly.

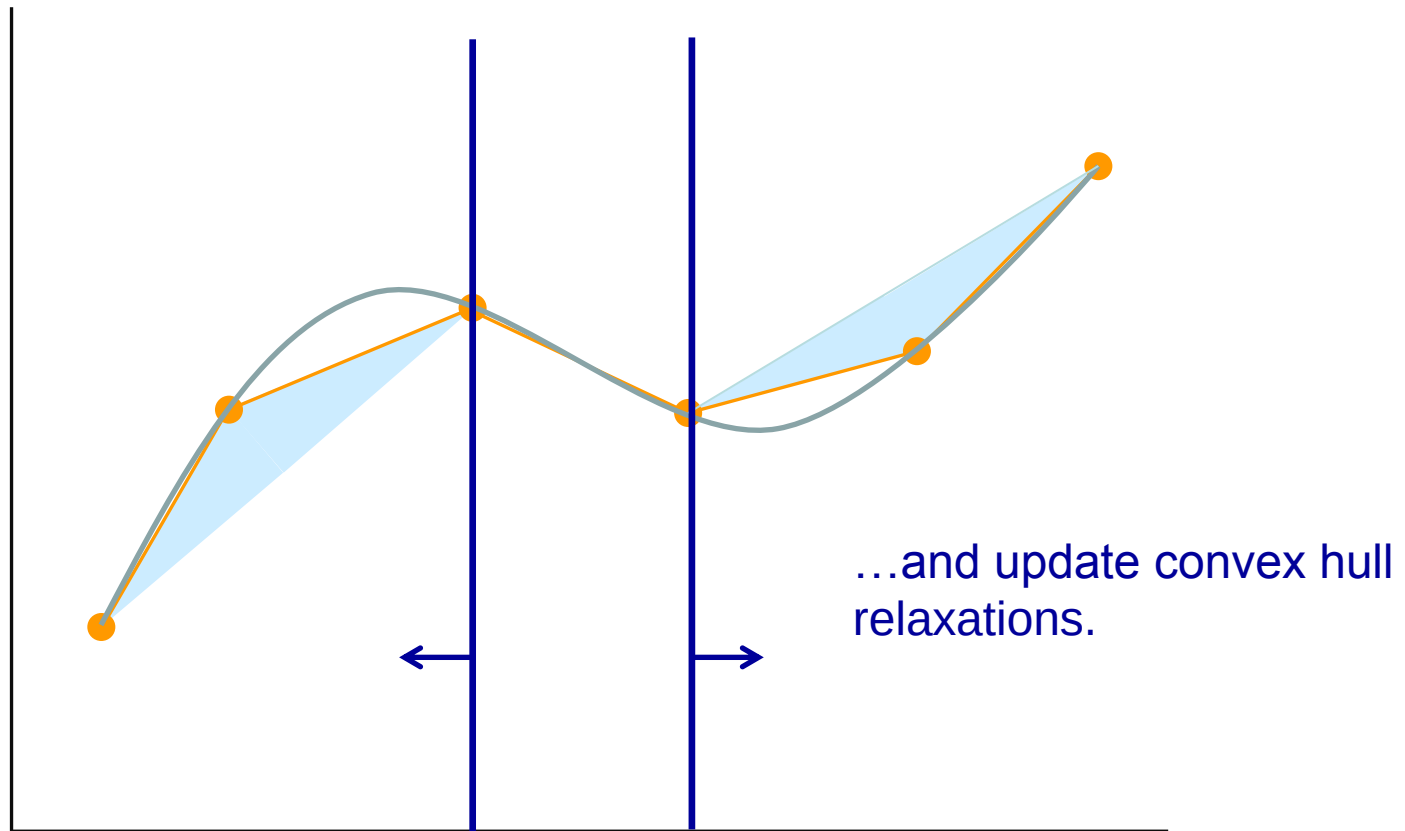
Piecewise linear functions



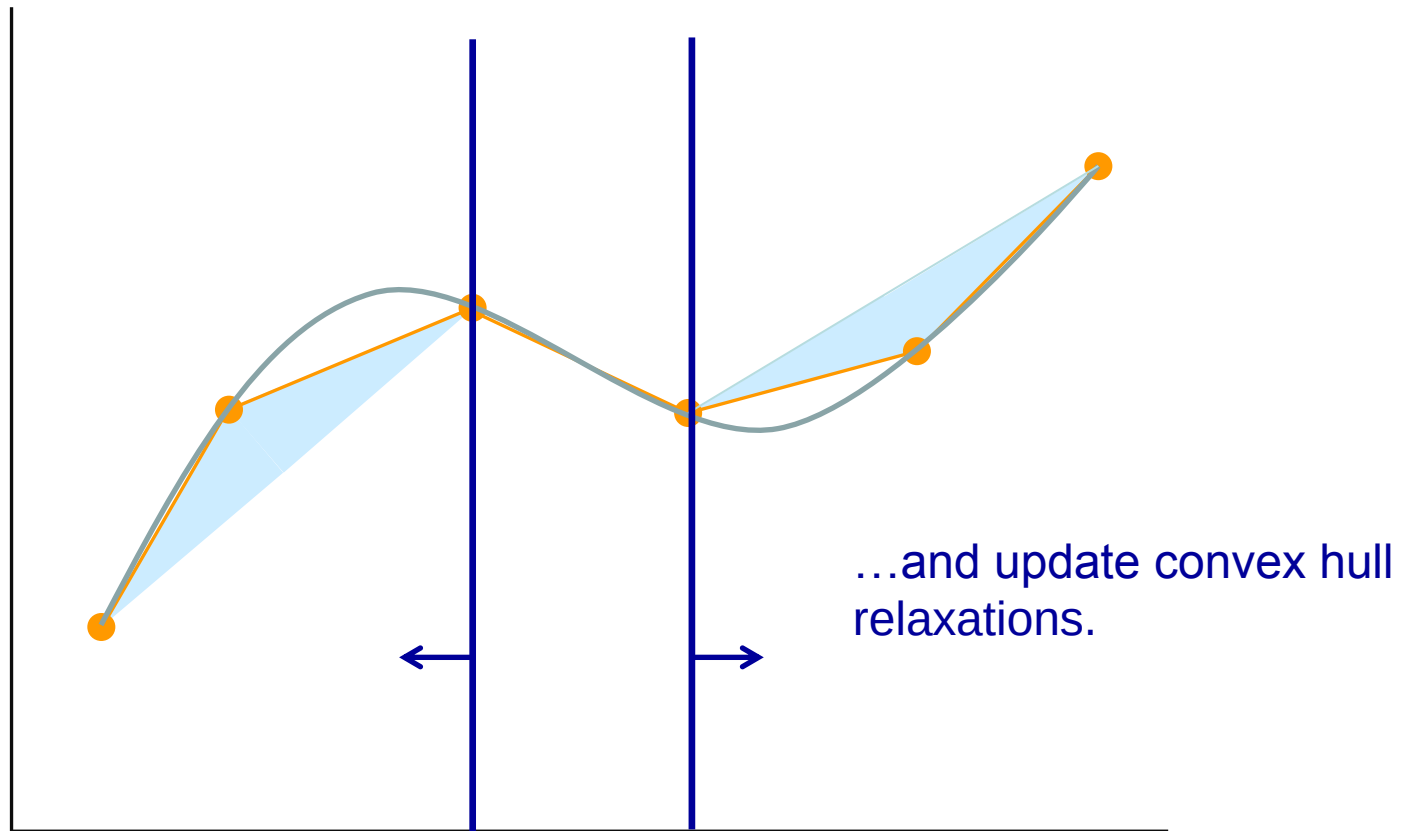
Piecewise linear functions



Piecewise linear functions



Piecewise linear functions



Easily extended to functions $f_j(x_i, x_k)$
By computing 3D convex hull.

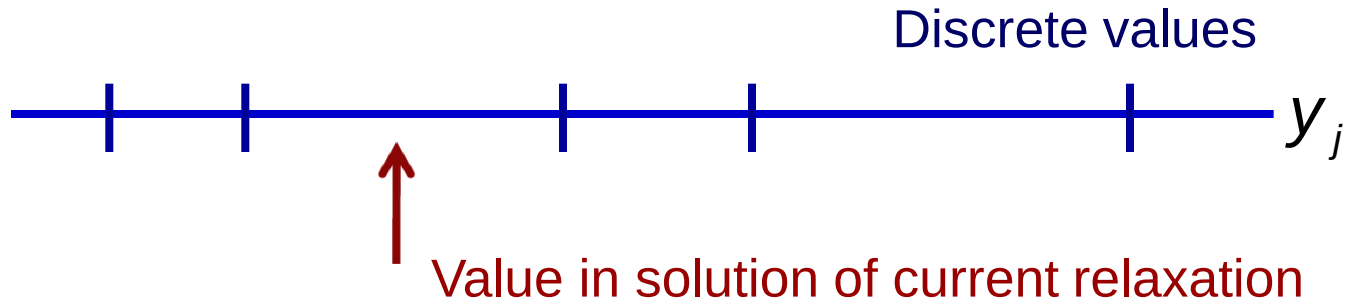
Branching

- In general, branch on discrete values of a variable.
 - ...rather than introduce 0-1 variables to model discrete values.
- For a troublesome continuous variable, **discretize it** and branch.
 - Use **many break points** without increasing size of model.

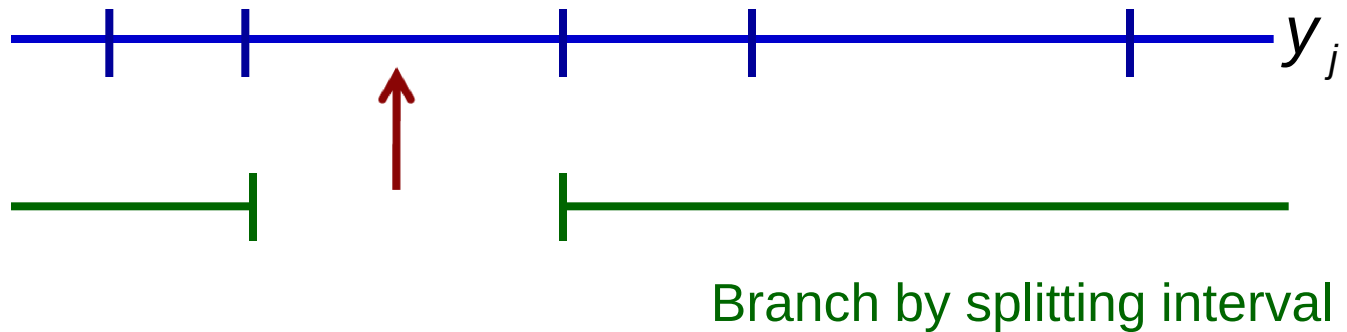
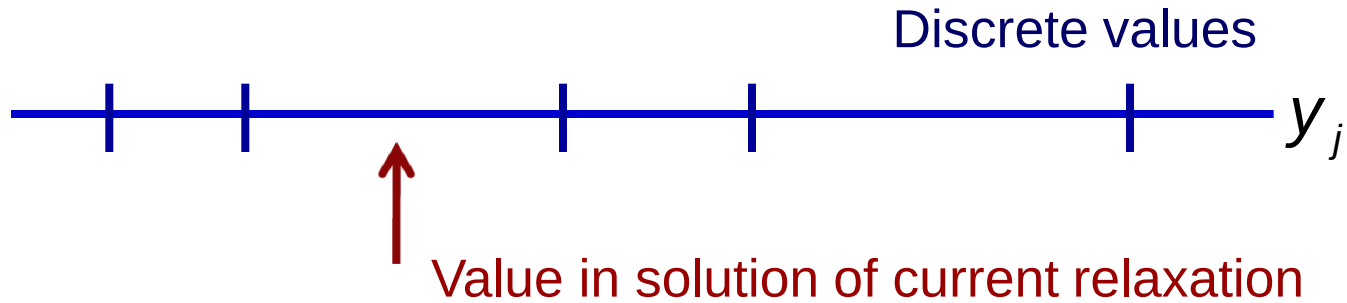
Branching

- In general, branch on discrete values of a variable.
 - ...rather than introduce 0-1 variables to model discrete values.
- For a troublesome continuous variable, **discretize it** and branch.
 - Use **many break points** without increasing size of model.
- This may allow for a **convex** “relaxation” (actually, **quasi-relaxation**)
 - If the model becomes convex when discretized variables are fixed.
 - A quasi-relaxation is not a valid relaxation but yields a valid bound on the objective function.

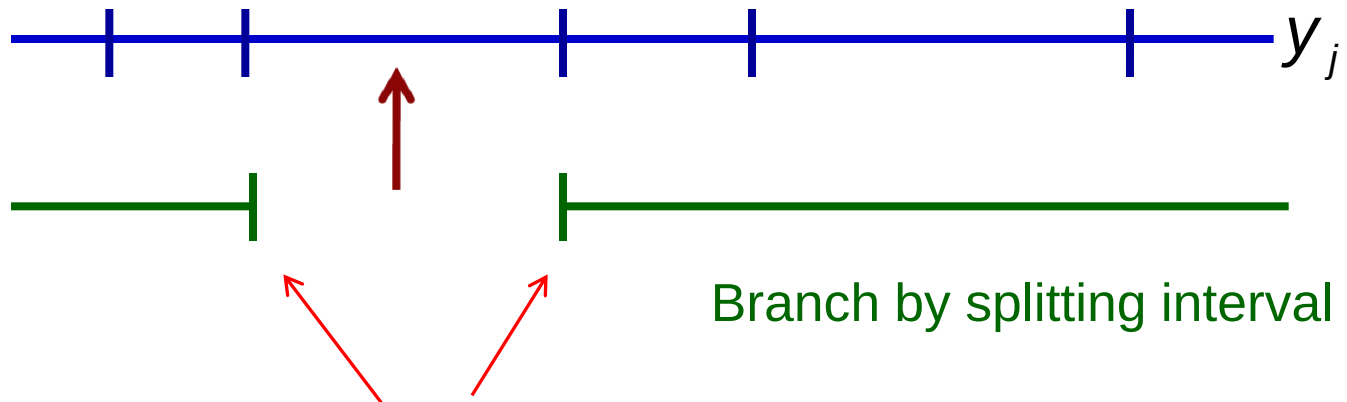
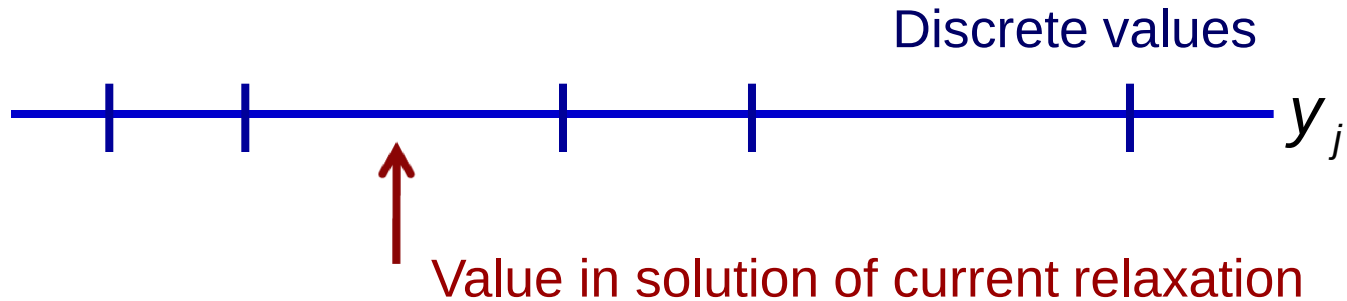
Branching



Branching



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.

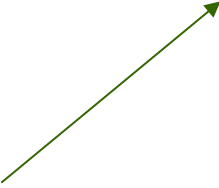
Quasi-relaxation

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

Each g^j is
a vector of
functions



Quasi-relaxation

Consider the problem $\min f(x)$

$$\begin{array}{l} g^j(x, y_j) \leq 0, \text{ all } j \\ x \in \mathbb{R}^n, y_j \text{ discrete} \end{array}$$

Each g^j is
a vector of
functions

Each y_j is
a scalar
variable

Quasi-relaxation

Consider the problem $\min f(x)$

$$\begin{array}{l} g^j(x, y_j) \leq 0, \text{ all } j \\ x \in \mathbb{R}^n, y_j \text{ discrete} \end{array}$$

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 $\min f(x)$

$$\begin{aligned} & g^j(x, y_j) \leq 0, \text{ all } j \\ & x \in \mathbb{R}^n, y_j \text{ discrete} \end{aligned}$$

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 $\min f(x)$

$$\begin{aligned} & g^j(x, y_j) \leq 0, \text{ all } j \\ & x \in \mathbb{R}^n, y_j \text{ discrete} \end{aligned}$$

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

Theorem (JNH)

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

$$\begin{aligned} &\min f(x) \\ &g(x^1, y_L) + g(x^2, y_U) \leq 0 \\ &\alpha x^L \leq x^1 \leq \alpha x^U \\ &(1 - \alpha)x^L \leq x^2 \leq (1 - \alpha)x^U \\ &x = x^1 + x^2, \alpha \in [0, 1] \end{aligned}$$

Quasi-relaxation

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

Theorem (JNH)

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

$$\begin{aligned} \min f(x) \\ g(x^1, y_L) + g(x^2, y_U) &\leq 0 \\ \alpha x^L &\leq x^1 \leq \alpha x^U \\ (1 - \alpha)x^L &\leq x^2 \leq (1 - \alpha)x^U \\ x &= x^1 + x^2, \alpha \in [0, 1] \end{aligned}$$

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

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

Quasi-relaxation

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

Theorem (JNH)

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

$$\begin{aligned} &\min f(x) \\ &g(x^1, y_L) + g(x^2, y_U) \leq 0 \\ &\alpha x^L \leq x^1 \leq \alpha x^U \\ &(1-\alpha)x^L \leq x^2 \leq (1-\alpha)x^U \\ &x = x^1 + x^2, \alpha \in [0, 1] \end{aligned}$$

Quasi-relaxation

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

Theorem (JNH)

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

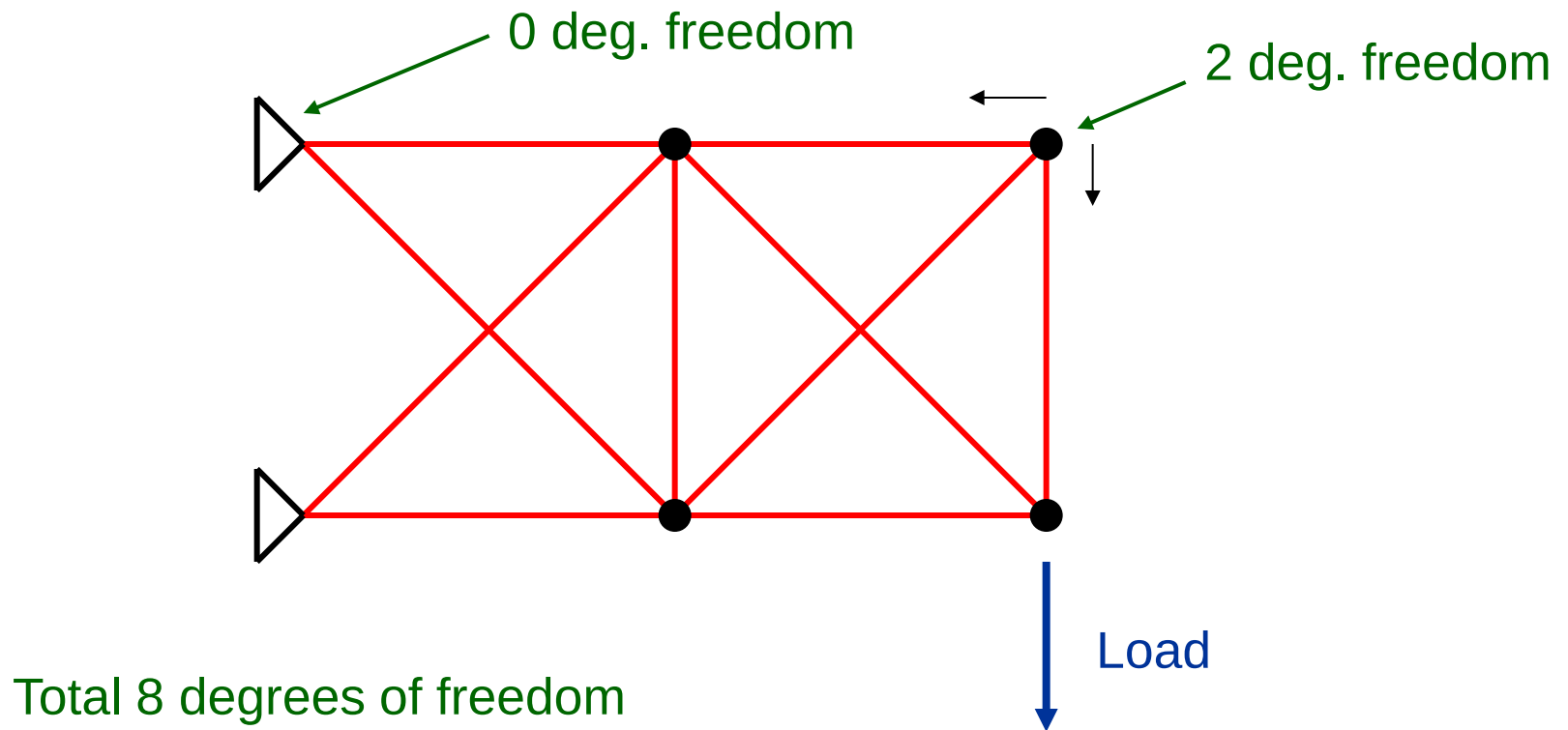
$$\begin{aligned} &\min f(x) \\ &g(x^1, y_L) + g(x^2, y_U) \leq 0 \\ &\alpha \boxed{x^L} \leq x^1 \leq \alpha \boxed{x^U} \\ &(1-\alpha)x^L \leq x^2 \leq (1-\alpha)x^U \\ &x = x^1 + x^2, \quad \alpha \in [0, 1] \end{aligned}$$

Bounds on x

Example: 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



Truss Structure Design

$$\begin{aligned} \min \quad & \sum_i h_i A_i \\ \text{s.t.} \quad & \sum_i \cos \theta_{ij} s_i = p_j, \text{ all } j \\ & \sum_j \cos \theta_{ij} d_j = v_i, \text{ all } i \end{aligned}$$

Nonlinear,
nonconvex

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

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

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

$$V_k (A_i = A_{ik})$$

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

Truss Structure Design

Can convert to MILP model by introducing new variables.

$$\begin{aligned} \min \quad & \sum_i h_i \sum_k A_{ik} y_{ik} \\ \text{s.t.} \quad & \sum_i \cos \theta_{ij} s_i = p_j, \text{ all } j \\ & \sum_j \cos \theta_{ij} d_j = \sum_k v_{ik}, \text{ all } i \\ & \frac{E_i}{h_i} \sum_k A_{ik} v_{ik} = s_i, \text{ all } i \\ & v_i^L \leq v_i \leq v_i^U, \text{ all } i \\ & d_j^L \leq d_j \leq d_j^U, \text{ all } j \\ & \sum_k y_{ik} = 1, \text{ all } i \end{aligned}$$

0-1 variables indicating size of bar i

Elongation variable disaggregated by bar size

Hooke's law becomes linear

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 $g(x, y_j) = 0$ with g semihomogeneous in x and concave (linear) in y_j because we can write it

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

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

Truss Structure Design

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

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

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

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

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

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

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

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

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

Hooke's law is linearized

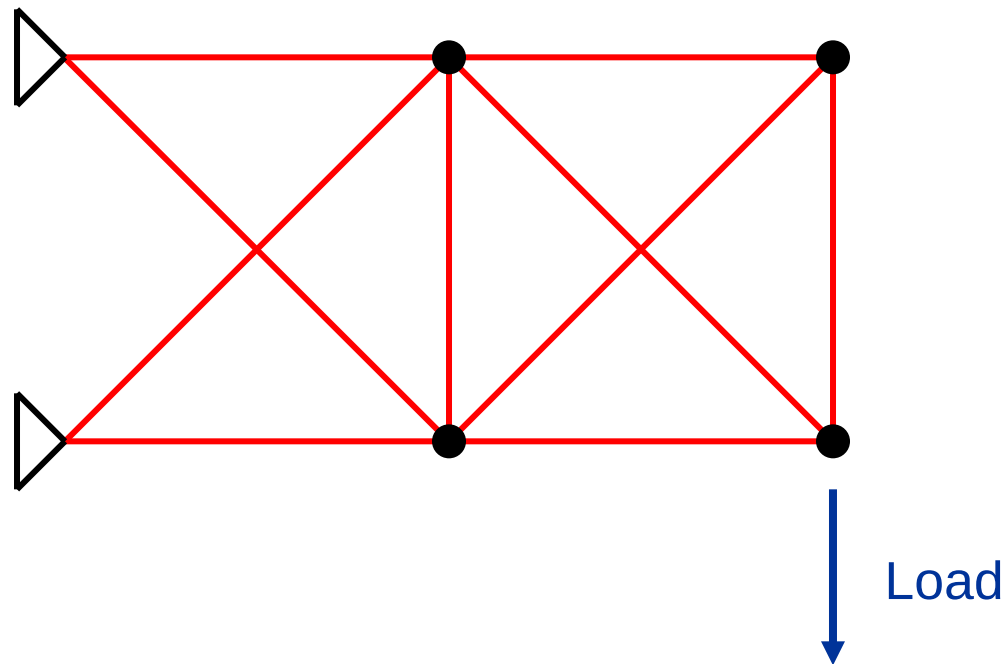
Elongation bounds split into 2 sets of bounds

Truss Structure Design

Some computational results...

10-bar cantilever truss

Yunes, Aron, JNH (2010),
based on
Bollapragada, Ghattas, JNH (2001)



Truss Structure Design

SIMPL = integrated solver that implements CP-style branching and quasi-relaxations

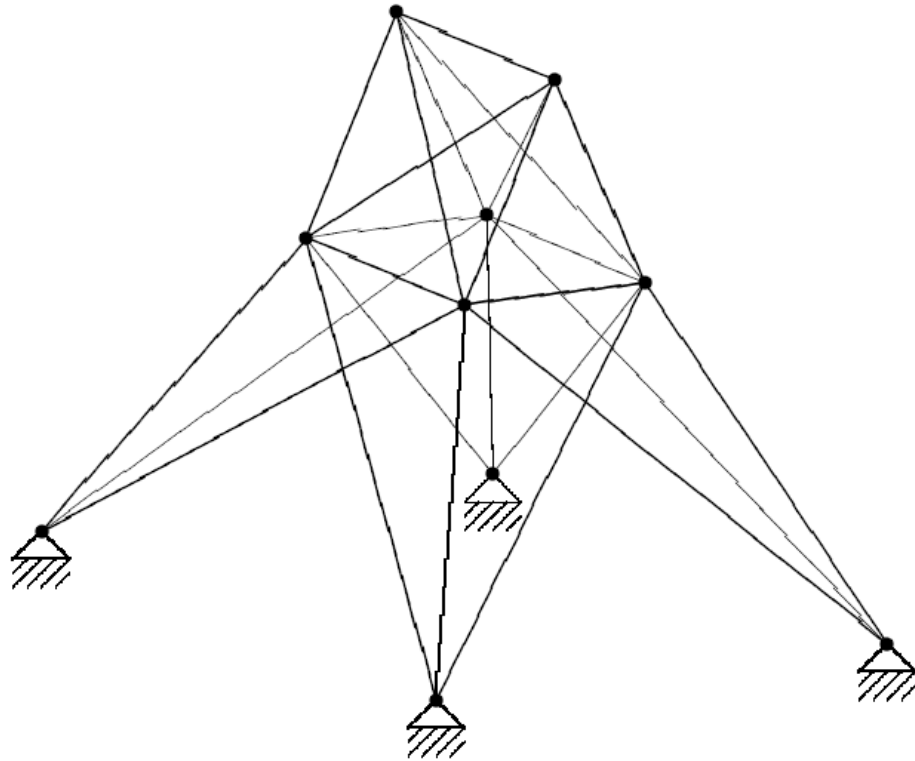


Computational results (seconds)

No. bars	Loads	BARON	CPLEX	SIMPL
10	1	5.3	0.40	0.08
10	1	3.8	0.26	0.07
10	1	8.1	0.83	0.49
10	1	8.8	1.2	0.63
10	2	24	4.9	1.84
10	2*	327	146	65
10	2*	2067	1087	651

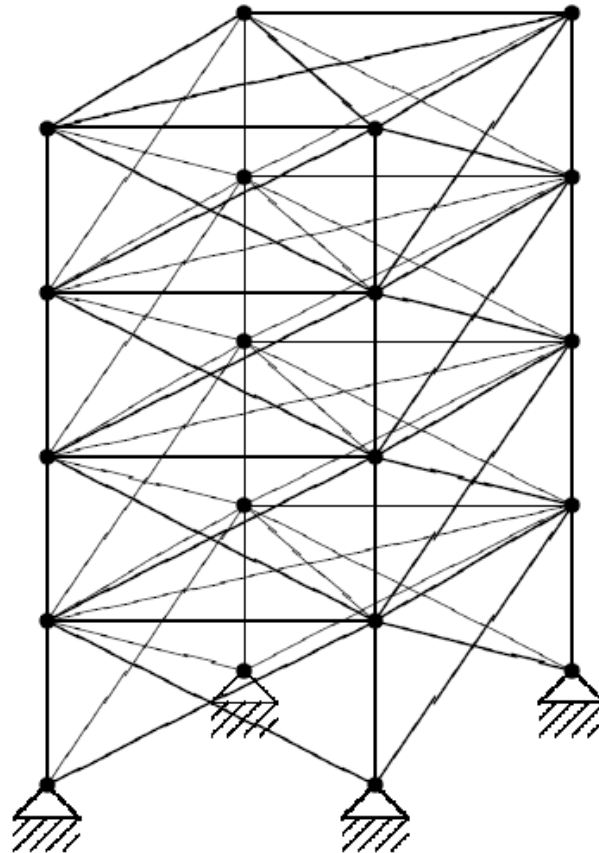
Truss Structure Design

25-bar problem



Truss Structure Design

72-bar problem



Truss Structure Design

Computational results (seconds)

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

* no feasible solution found

** best feasible solution has cost 32,700

Decision diagrams

- A **decision diagram** can represent the feasible set of a discrete optimization problem.
 - An optimal solution is a **shortest path** in the diagram.
 - Linearity, convexity **irrelevant**.
 - Provide **enhanced propagation** in a CP context.
- Proposal: **discretize** continuous variables and optimize over a decision diagram.
 - Branching in **relaxed** decision diagrams may permit **massive discretization**.
 - A “big data” technique.

Decision diagrams

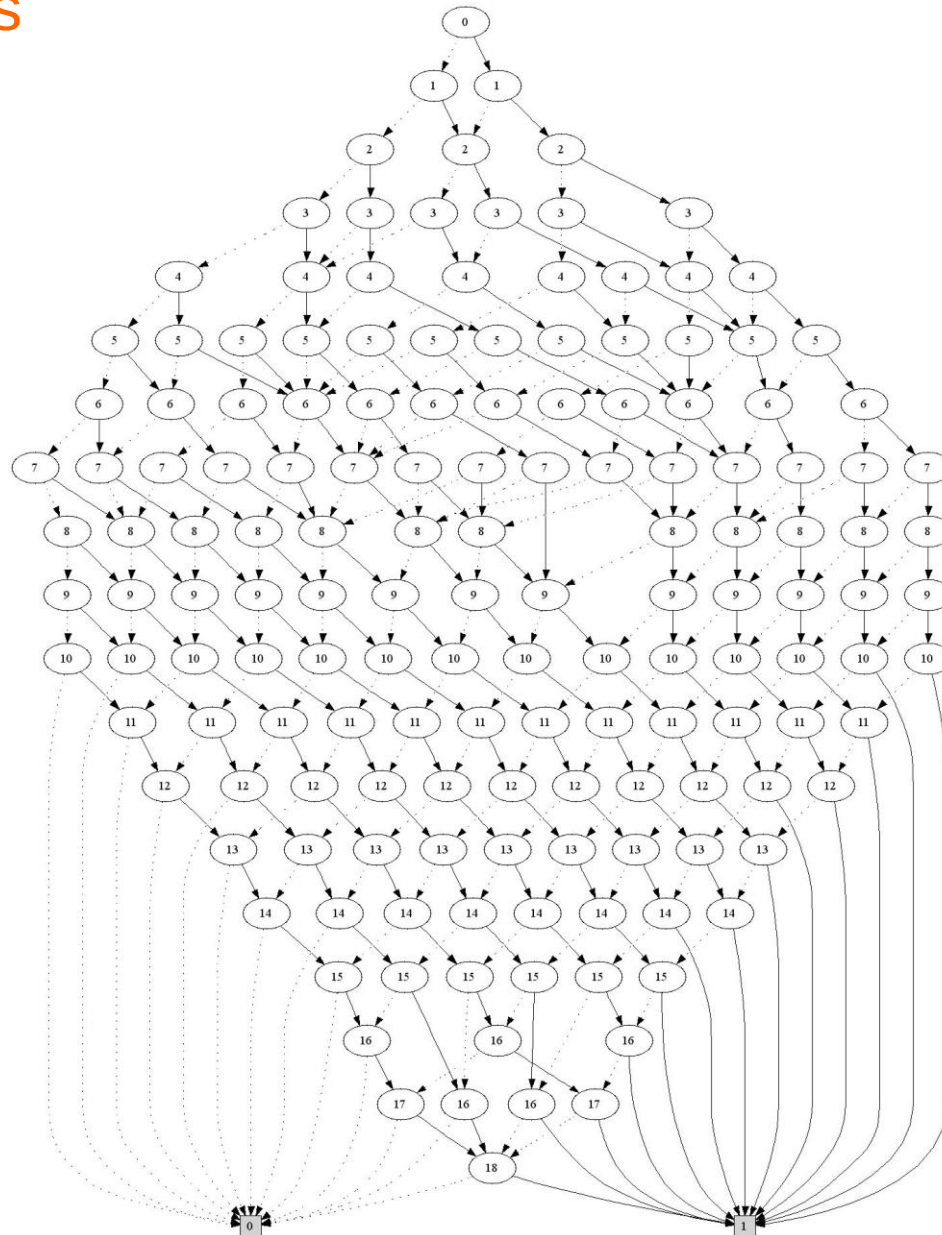
- The knapsack constraint

$$300x_0 + 300x_1 + 285x_2 + 285x_3 + 265x_4 + 265x_5 + 230x_6 + 23x_7 + 190x_8 + 200x_9 + \\ 400x_{10} + 200x_{11} + 400x_{12} + 200x_{13} + 400x_{14} + 200x_{15} + 400x_{16} + 200x_{17} + 400x_{18} \geq 2701$$

has 117,520 minimal feasible solutions.

- But its reduced decision diagram has only 152 nodes...

Decision diagrams

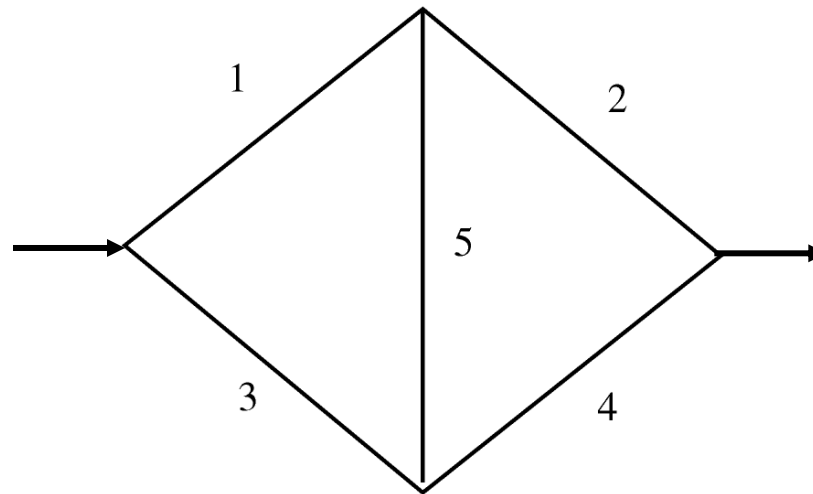


A branch from layer i represents fixing x_i to 0 (dashed arc) or 1 (solid arc).

Paths to 1 correspond to feasible solutions.

Example: network reliability

- Minimize cost subject to a bound on reliability (highly nonconvex)
 - System of 5 bridges:



$$R = R_1 R_2 + (1 - R_2) R_3 R_4 + (1 - R_1) R_2 R_3 R_4 + R_1 (1 - R_2) (1 - R_3) R_4 R_5 + (1 - R_1) R_2 R_3 (1 - R_4) R_5$$

Example: network reliability

The problem:

$$\min \sum_j c_j \boxed{x_j} \quad \leftarrow \text{Number of links at bridge } j$$

$$R \geq R_{\min}$$

$$R = R_1 R_2 + (1 - R_2) R_3 R_4 + (1 - R_1) R_2 R_3 R_4 \\ + R_1 (1 - R_2) (1 - R_3) R_4 R_5 + (1 - R_1) R_2 R_3 (1 - R_4) R_5$$

$$R_j = 1 - (1 - \boxed{r_j})^{x_j}, \text{ all } j$$

$$x_j \in \{0, 1, 2, 3\} \quad \leftarrow \text{Reliability of one link for bridge } j$$

Set min desired reliability to $R_{\min} = 60\%$

Eliminate variables R_i , leaving one **continuous** variable R .
Discretize R for the decision diagram.

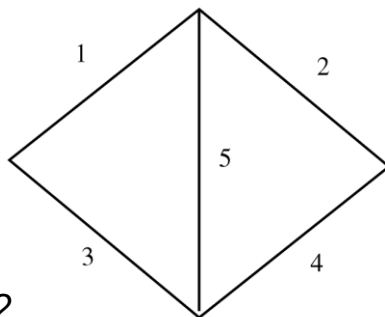
Example: network reliability

Optimal solution

Decision diagram has
308 nodes, generated
in 1.1 sec.

Computing optimal solution
is trivial (shortest path).

Bonus: we get complete
postoptimality analysis
from decision diagram



$c_{opt} + \Delta$	x_1	x_2	x_3	x_4	x_5	R
50:	0	0	1	1	0	72
60:	1	1	0	0,2		79
85:	2					84
90:			2	3		86
95:		2			1	88
100:						95
120:						97
125:	3					
155:		3			2	
160:						98
170:						99
180:			3			
230:					3	

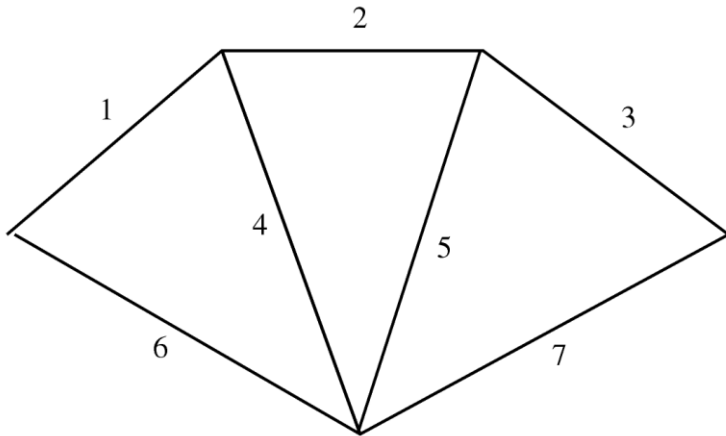
Hadzic and JNH (2006).

Example: network reliability

Nonlinear constraints are increasingly complex for larger networks.

Decision diagram has 1779 nodes, generated in 14.8 sec.

7 bridges

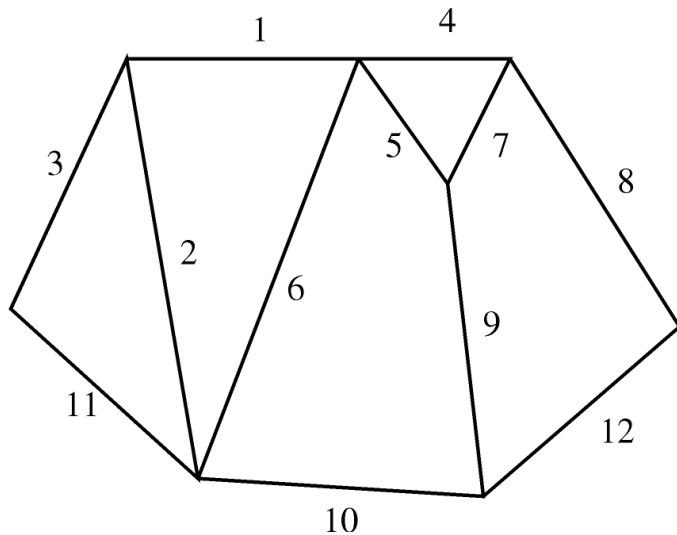


$c_{opt} + \Delta$	x_1	x_2	x_3	x_4	x_5	x_6	x_7	R
9:	0	0	0	0	0	1	1	72.2
11:			1		1		0	
12:	1			1		0		
13:		1				2		82.9
14:					2		2	
15:			2	2				
16:	2							
17:					3	3		84.6
18:		2		3				95.2
19:			3				3	
20:	3							
22:								97.2
23:		3						
27:								99.2
34:								99.4
40:								99.6
43:								99.7
47:								99.8
54:								99.9

Example: network reliability

Decision diagram has
69,457 nodes, generated
in 2933 sec.

12 bridges



$c_{opt} + \Delta$	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	R
180	1	0	2	3	0	0	0	2	0	0	0	0	80
185			3	2									82
190								3					
195													83
200										1			
205													86
210	2						1						
215													88
220										2			
225					1				1				
230	0		0			1,2					1		
235			1										
240		1		1			2			3	2	1	
250				0				0,1				2	
255													91
260	3								2				
265													93
270					2	3	3						
290											3		
300		2											
305									3				
310												3	
315					3								94
340													95
360		3											
365													96
380													97
430													98
485													99

Example: portfolio design

Expected yield rate

Number of blocks of security i purchased

$$\max \sum_{i=1}^n \mu_i x_i$$

Maximum variance

$$\sum_{i=1}^n \sum_{j=1}^n c_i c_j \sigma_{ij} x_i x_j \leq V_{max}$$

Maximum investment

$$\sum_{i=1}^n c_i x_i \leq W$$

Maximum number of securities in portfolio

$$\sum_{i=1}^n \delta(x_i) \leq K$$

1 if $x_i > 0$,
0 otherwise
(no need for
0-1 variables)

$$x_i \in D_i, \quad i = 1, \dots, n$$

Example: portfolio design

10 securities,
max 7 selected.

Decision diagram has
59,802 nodes, generated
in 63 sec.

Trivial to compute yield/risk
tradeoff.

Hadzic and JNH (2006).

$C_{opt} - \Delta$	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
21797	6	0	7	0	3	7	6	0	3	7
21754						6	7			
21705				1					0	6
21683									2	
21678	7		6							
21673					5					
21670					4					
21663				2	0					
21647								2		
21642							5		1	
21630					2					
21624								1,3		
21604					1					
21599						5				
21572			5							
21567									4	
21562										5
21532	5									
21529							4			
21484									5	4
21467						4				
21456								4		
21412					6					
21404							3			
21370		1								
21351								5		
21330									6	
21312			4							
21232						3				
21215										3
21134					7					
21133		2								

Decision diagrams

- What if there are **many continuous variables**?
 - Discretize them!
 - Use limited-width **relaxed** decision diagram to obtain optimization bounds.
 - Branch in relaxed decision diagram.
- So far, this method has been applied to IP:
 - Competitive with state-of-the-art IP solvers, or better.
 - Construction of relaxed decision diagram dynamically creates finer granularity for more promising discrete values.

Bergman, Cire, van Hoeve, JNH (2013)

McCormick factorization

- Can be managed with **global constraints + semantic typing**.

Cire, JNH, Yunes (2013).

Want to know more about CP and optimization?

- See this websites for links to tutorials (slides & videos):

<http://web.tepper.cmu.edu/jnh/slides.html>

- See also:

<http://moya.bus.miami.edu/~tallys/integrated.php> (CP + optimization)

<http://www.andrew.cmu.edu/user/vanhoeve/mdd/> (decision diagrams)