Integrating Solution Methods through Duality

John Hooker

US-Mexico Workshop on Optimization
Oaxaca, January 2011

Zapotec
Duality mask
Late classical
period

Represents
life/death,
day/night,
heat/cold

See a similar mask
in Centro Cultural
Santo Domingo

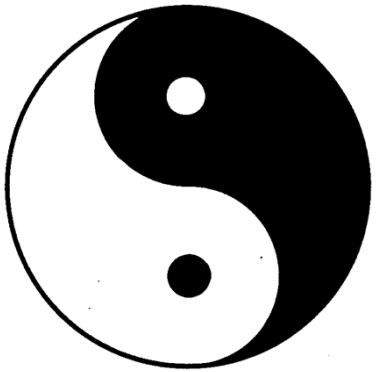
Duality

Two perspectives on the same phenomenon

Lalita Mahatripurasundari
representing Brahman-Atman consciousness (mind/body)

Complementarity

Mingling of opposites



Yin/yang
Daoism

Alharamazda
Zoroastrian god
Good/evil

Goal

- Design a **general-purpose** solver that exploits **problem-specific structure**.

Goal

- Design a **general-purpose** solver that exploits **problem-specific structure**.

- Approach:
  - Find **common algorithmic structure** across solution methods
  - Adjust specific elements of this structure to suit the problem.

Goal

- Design a **general-purpose** solver that exploits **problem-specific structure**.

- Approach:
  - Find **common algorithmic structure** across solution methods
  - Adjust specific elements of this structure to suit the problem.

- Thesis:
  - Successful combinatorial optimization methods use a **primal-dual-dual** solution strategy.
  - The duals can be designed to **exploit problem structure**.
  - This perspective can **unify methods** and lead to **new ones**.

Outline

- Two dualities
- Example: Integrated approach to IP
- Focus on inference duality
- Example: SAT
- Examples of integrated solving
  - Piecewise linear optimization
  - Planning and disjunctive scheduling
  - Planning and cumulative scheduling
  - Single-facility scheduling
  - Truss structure design (nonlinear)
- Conclusion

Surviving this talk

- We must cover a wide variety of problems to make the point.

- Focus on the **structural parallels**
  - Rather than the **details** of each problem

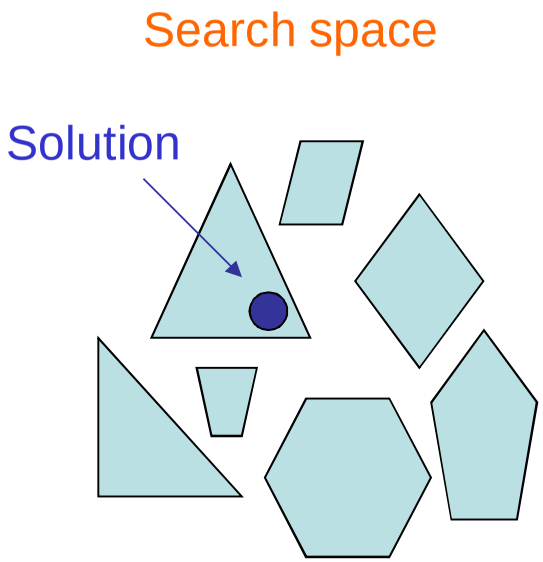
Two dualities

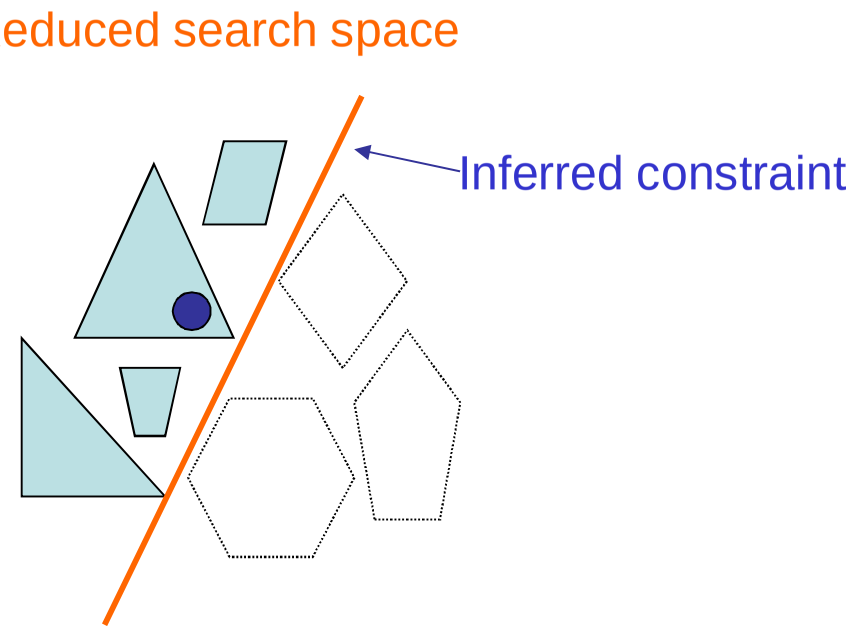
- **Inference** and **relaxation** can exploit problem structure.
  - They work together with search in most combinatorial optimization methods.

- Search, inference, and relaxation are linked by two underlying **dualities**:
  - Duality of **search and inference**
  - Duality of **search and relaxation**

Two ways to exploit structure

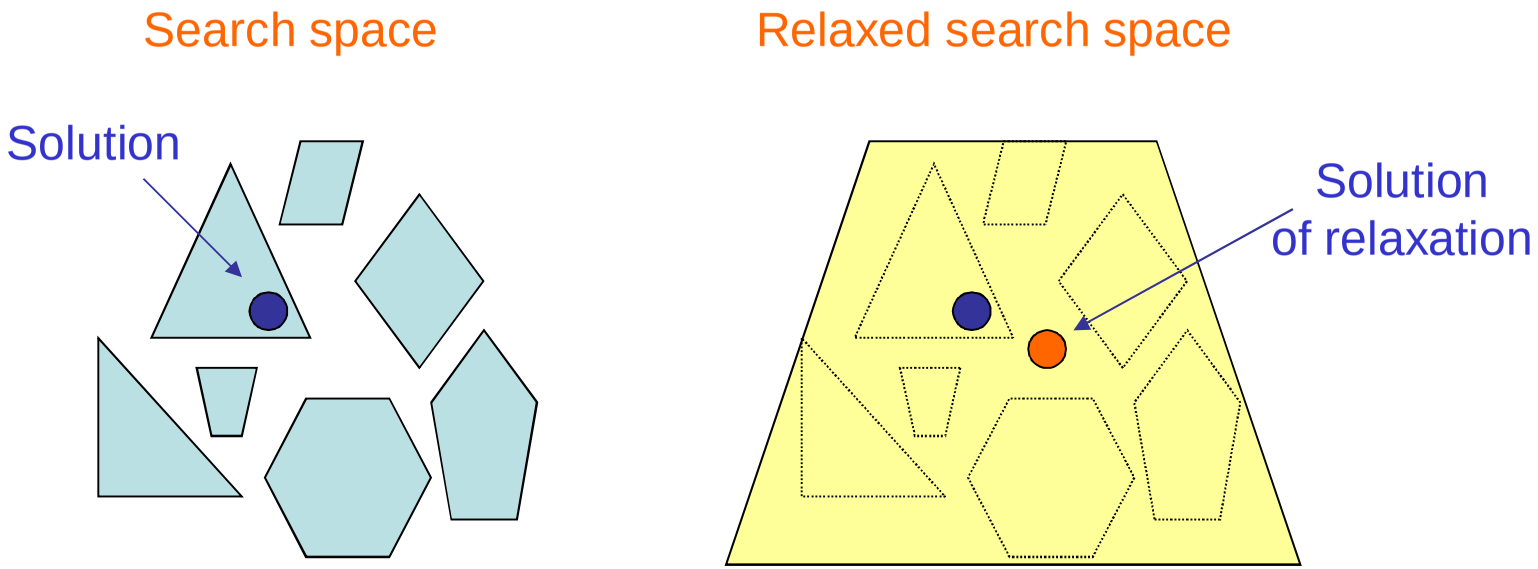
- **Inference**
  - Use knowledge of problem structure to reveal **hidden information**.
  - This can exclude **unpromising** areas of the search space..





Two ways to exploit structure

- **Relaxation**
  - Use knowledge of problem structure to design a **larger but simpler** search space.
  - Solution of relaxation may be **near** solution of original problem and provides a **bound**.



Two dualities

- Duality of search and inference.
  - **Search** looks for a certificate of **feasibility**.
  - The **inference dual** looks for a certificate (proof) of **infeasibility** (or **optimality**).

- Duality of search and relaxation.
  - **Search** enumerates **restrictions** of the problem.
  - The relaxation dual enumerates (parameterized) **relaxations** of the problem.

Primal-dual-dual methods

- Primal methods.
  - Enumerate possible **solutions** (in general, problem restrictions).
- Dual methods.
  - Enumerate **proofs** or **relaxations**.
- Primal-dual methods.
  - Solve the **primal** and a **dual** simultaneously.
- Primal-dual-dual methods.
  - Solve the primal and **both duals** simultaneously.
  - Strategy of **most successful combinatorial optimization methods**.

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, Lagrangean
- 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, dynamic backtracking, etc.
  - **Inference:** Conflict clauses
  - **Relaxation:** Same as restriction
- Local search
  - **Search:** Enumerate neighborhoods.
  - **Inference:** Tabu list, etc.
  - **Relaxation:** Same as restriction

Example: Integrated approach to IP

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$x_i \in \{0,1,2,3\}$$

Inference: Bounds propagation

Simple constraint programming technique

$$\min 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$x_i \in \{0,1,2,3\}$$

$$x_1 \geq \left\lceil \frac{42 - 5 \cdot 3 - 4 \cdot 3 - 3 \cdot 3}{7} \right\rceil = 1$$

Bounds propagation

Simple constraint programming technique

$$\min 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$x_1 \in \{1,2,3\}, \quad x_2, x_3, x_4 \in \{0,1,2,3\}$$

Reduced domain

$$x_1 \geq \left\lceil \frac{42 - 5 \cdot 3 - 4 \cdot 3 - 3 \cdot 3}{7} \right\rceil = 1$$

Bounds propagation

Simple constraint programming technique

$$\min 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$x_1 \in \{1,2,3\}, \quad x_2, x_3, x_4 \in \{0,1,2,3\}$$

Reduced domain

$$x_1 \geq \left\lceil \frac{42 - 5 \cdot 3 - 4 \cdot 3 - 3 \cdot 3}{7} \right\rceil = 1$$

In general:

Bounds propagation aims for **bounds consistency**

Domain filtering aims for **hyperarc consistency** (GAC)

Relaxation: Linear programming

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

Replace domains with bounds

Inference: cutting planes (valid inequalities)

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$
$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$
$$x_1 + x_2 + x_3 + x_4 \leq 8$$
$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

We can create a **tighter** relaxation with the addition of **cutting planes**.

Inference: cutting planes (valid inequalities)

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

{1,2} is a **packing**

…because $7x_1 + 5x_2$ alone cannot satisfy the inequality, even with $x_1 = x_2 = 3$.

Inference: cutting planes (valid inequalities)

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

{1,2} is a **packing**

So, $4x_3 + 3x_4 \geq 42 - (7 \cdot 3 + 5 \cdot 3)$

General integer knapsack cut

which implies

$$x_3 + x_4 \geq \left\lceil \frac{42 - (7 \cdot 3 + 5 \cdot 3)}{\max\{4,3\}} \right\rceil = 2$$

Cutting planes (valid inequalities)

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

Maximal Packings	**Knapsack cuts**
{1,2}	$x_3 + x_4 \geq 2$
{1,3}	$x_2 + x_4 \geq 2$
{1,4}	$x_2 + x_3 \geq 3$

Knapsack cuts corresponding to nonmaximal packings can be nonredundant.

Continuous relaxation with cuts

$$\min\ 90x_1 + 60x_2 + 50x_3 + 40x_4$$

$$7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$$

$$x_1 + x_2 + x_3 + x_4 \leq 8$$

$$0 \leq x_i \leq 3, \quad x_1 \geq 1$$

$$x_3 + x_4 \geq 2$$

$$x_2 + x_4 \geq 2$$

$$x_2 + x_3 \geq 3$$

Knapsack cuts

Optimal value of 523.3 is a lower bound on optimal value of original problem.

Primal-dual-dual method

Search: branching

Inference: Bounds propagation, cutting planes

Relaxation: LP, domain store

Primal-dual-dual method

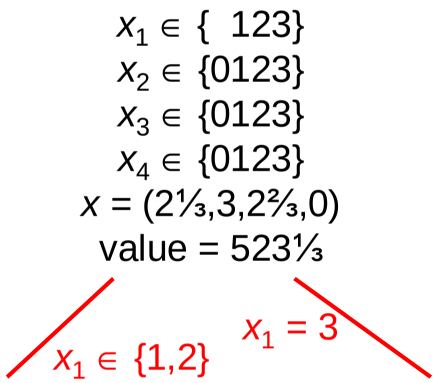
$x_1 \in \{\ 123\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (2\frac{1}{3}, 3, 2\frac{2}{3}, 0)$
value = $523\frac{1}{3}$

Propagate bounds and solve relaxation.

Since solution of relaxation is infeasible, branch.

Primal-dual-dual method

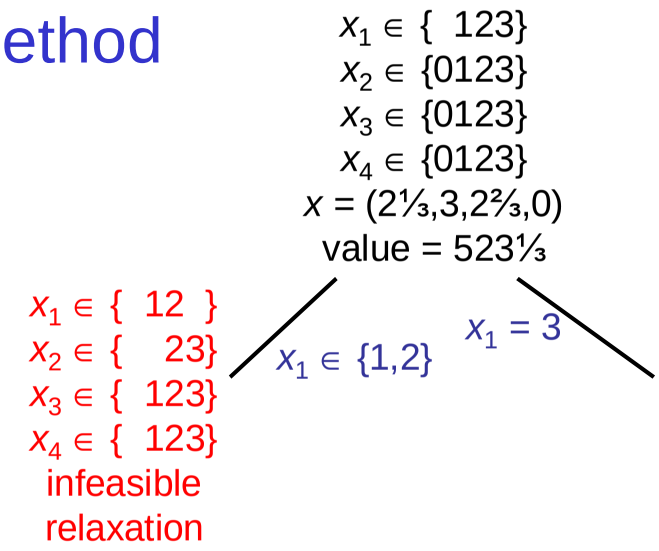
Branch on a variable with nonintegral value in the relaxation.



Primal-dual-dual method

Propagate bounds and solve relaxation.

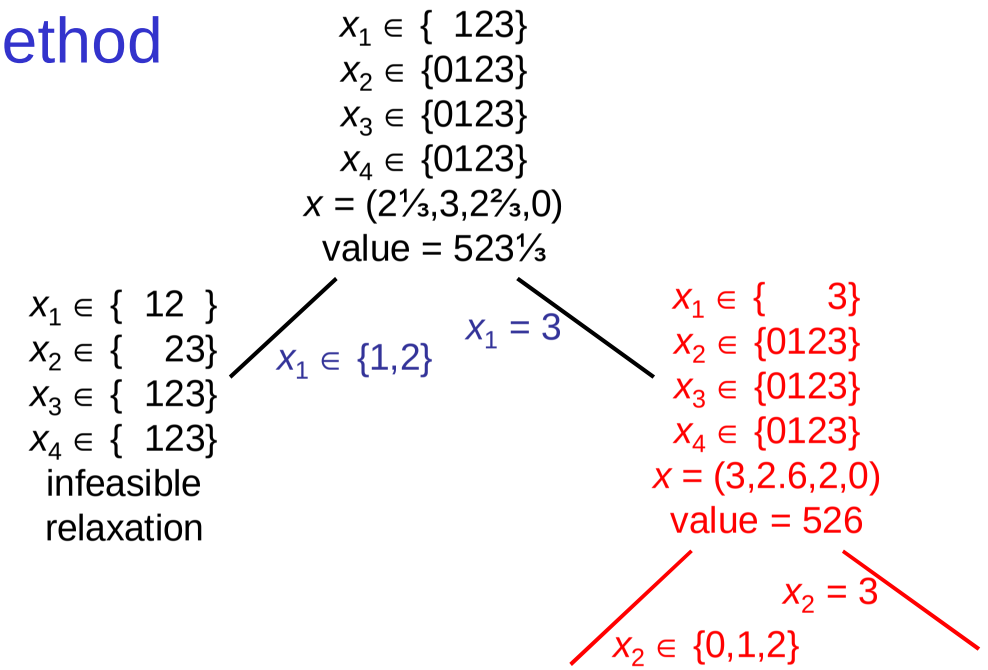
Since relaxation is infeasible, backtrack.



Primal-dual-dual method

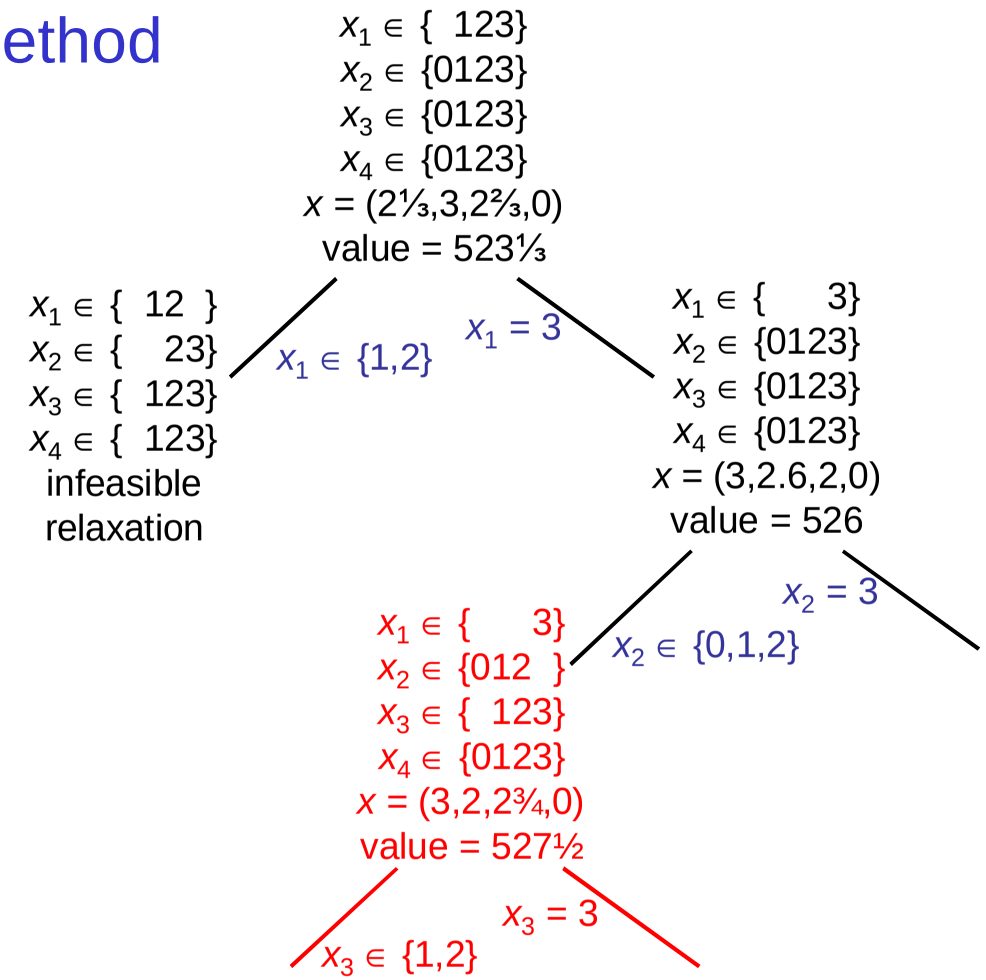
Propagate bounds and solve relaxation.

Branch on nonintegral variable.



Primal-dual-dual method

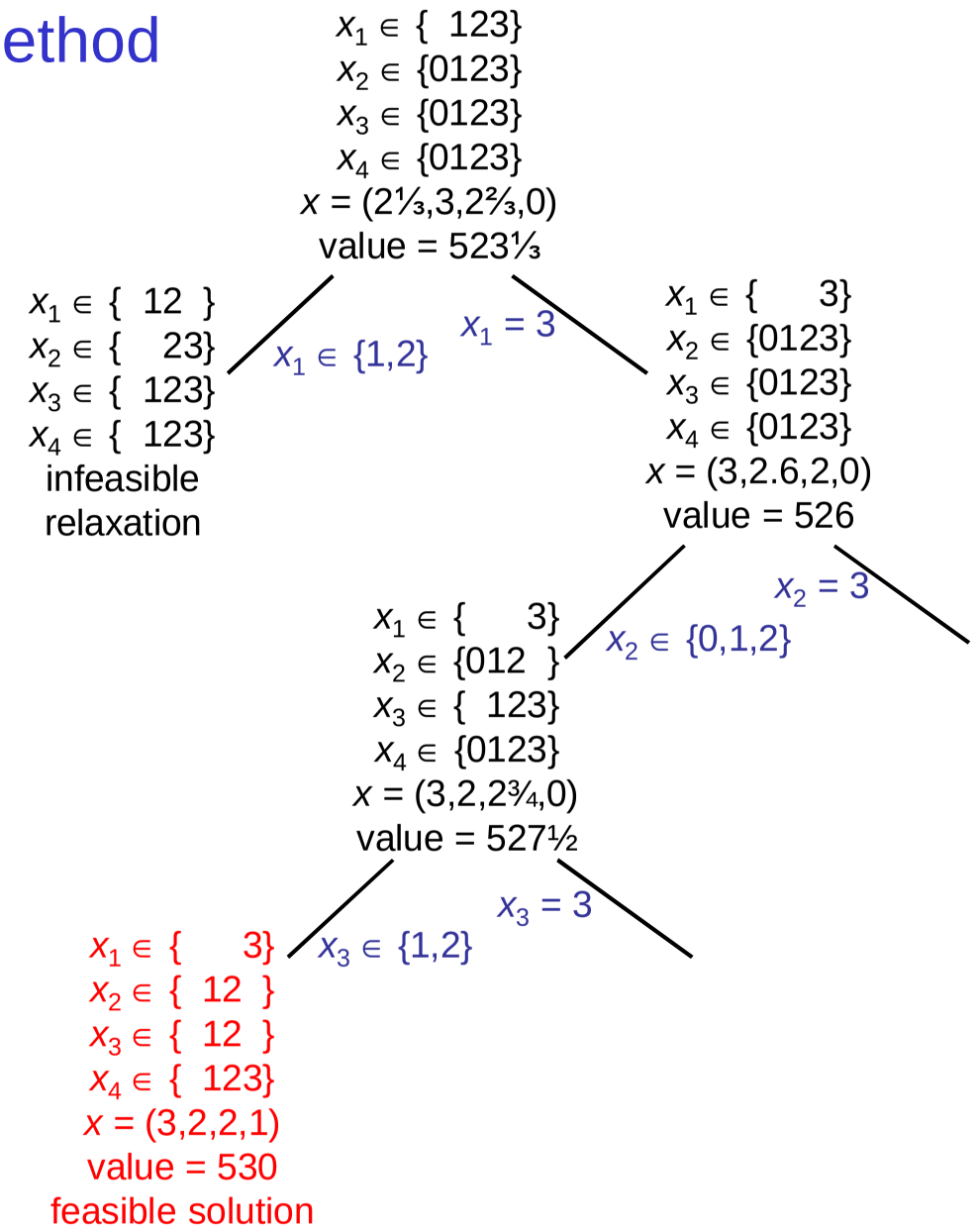
Branch again.



Primal-dual-dual method

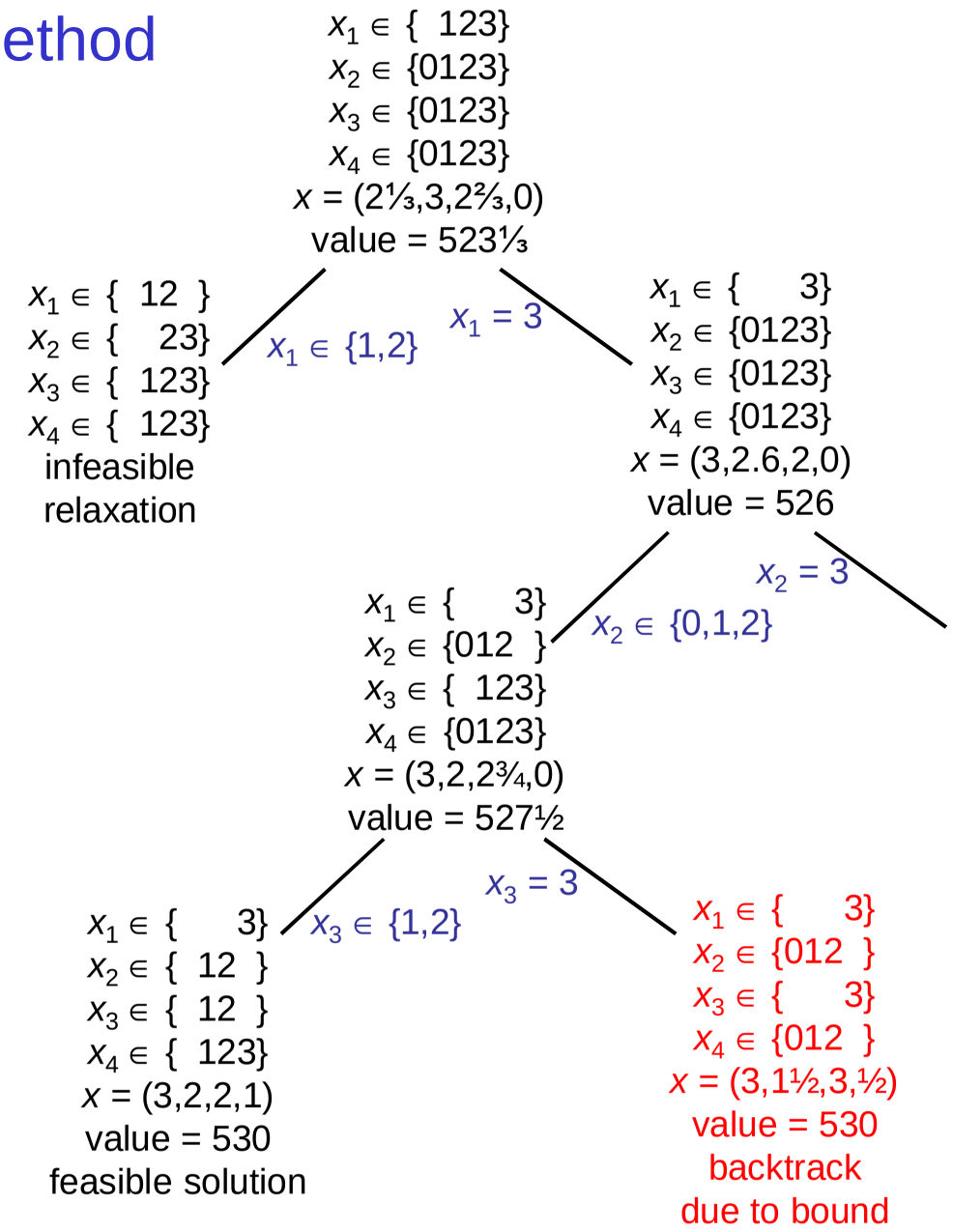
Solution of relaxation is integral.

This becomes the **incumbent solution**.



Primal-dual-dual method

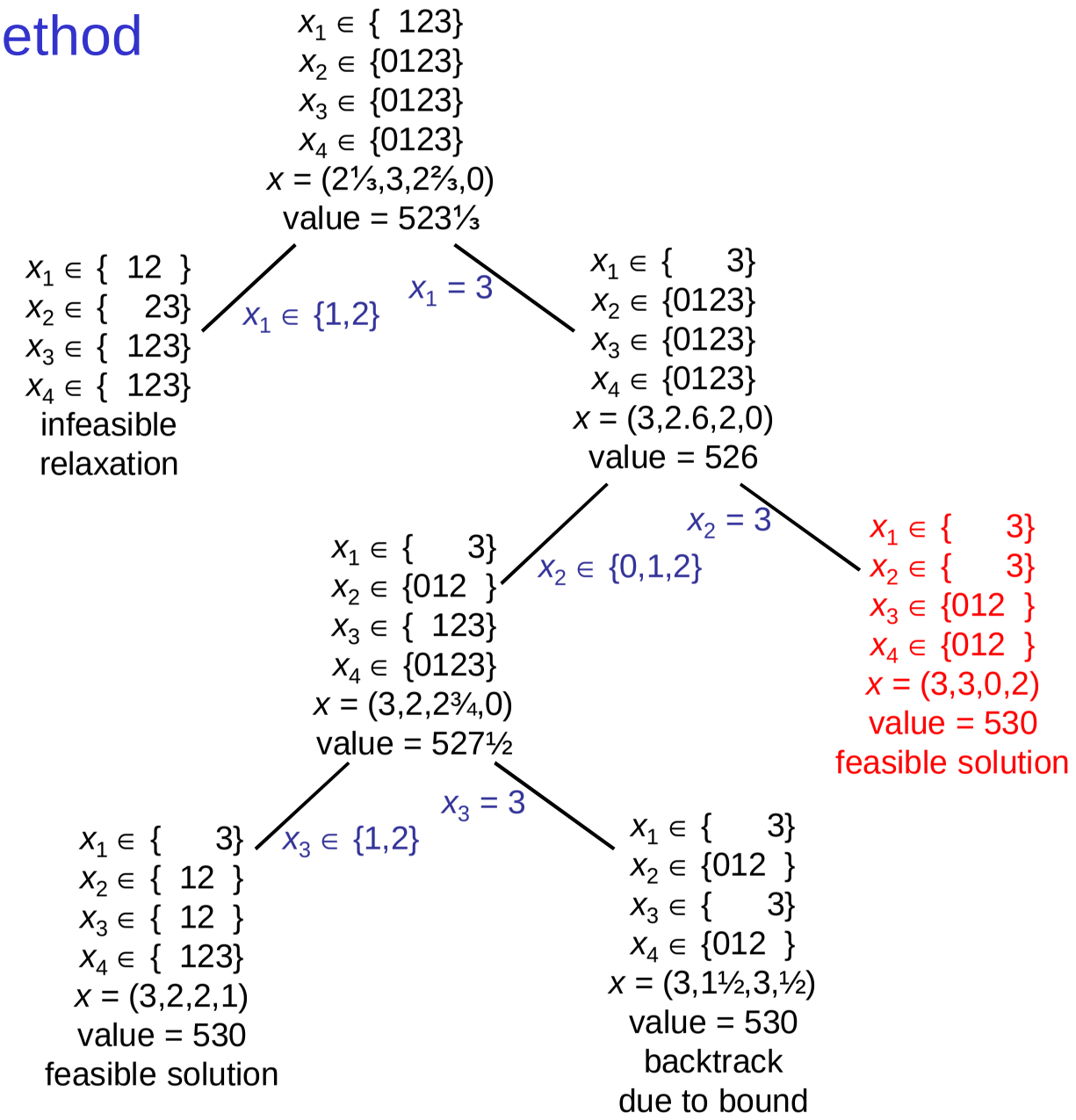
Solution is nonintegral, but use bound to backtrack,



Primal-dual-dual method

Another feasible solution found.

Search terminates with incumbent as optimum.



Example of search/inference duality

- We solved the **inference dual** by solving the **primal** (search) problem.
  - From top down, the tree is a **search**.
  - From bottom up, the tree is a **proof** of optimality.

Example of search/inference duality

- We solved the **inference dual** by solving the **primal** (search) problem.
  - From top down, the tree is a **search**.
  - From bottom up, the tree is a **proof** of optimality.
  - Cutting planes, domain filtering are steps in the proof.

Example of search/inference duality

- We solved the **inference dual** by solving the **primal** (search) problem.
  - From top down, the tree is a **search**.
  - From bottom up, the tree is a **proof** of optimality.
  - Cutting planes, domain filtering are steps in the proof.
  - The tree also encodes a solution of a **relaxation dual** (subadditive dual).

Example of search/inference duality

- We solved the **inference dual** by solving the **primal** (search) problem.
  - From top down, the tree is a **search**.
  - From bottom up, the tree is a **proof** of optimality.
  - Cutting planes, domain filtering are steps in the proof.
  - The tree also encodes a solution of a **relaxation dual** (subadditive dual).
- We can also solve the **primal** (search) problem by solving the inference dual
  - Inference dual generates nogoods for **nogood-based search**.

Relaxation and inference duality

- All(?) optimization duals are instances of both.

Outline:

- Relaxation dual
  - Example: Surrogate dual
  - Example: Lagrangean dual
- Inference dual
  - Example: LP dual
  - Example: Lagrangean dual
  - Example: Surrogate dual

Relaxation dual

Primal

$$\min f(x)$$
$$x \in S$$

Feasible set

Dual

$$\max \theta(u)$$
$$u \in U$$

Problem with relaxed epigraph, parameterized by u

where

$$\theta(u) = \min \{ f(x,u) \mid x \in S(u) \}$$

$$S(u) \supset S, \quad f(x,u) \le f(x), \quad \text{all } x \in S$$

Example: Surrogate dual

Primal

$$\min f(x)$$
$$g(x) \geq 0$$
$$x \in S$$

Dual

$$\max \theta(u)$$
$$u \geq 0$$

where

$$\theta(u) = \min \{ f(x) \mid ug(x) \geq 0, x \in S \}$$

- Replace constraints $g(x) \geq 0$ with a surrogate $ug(x) \geq 0$, objective function unchanged.
- LP dual is a special case.

Example: Lagrangean dual

Primal

$$\min f(x)$$
$$g(x) \geq 0$$
$$x \in S$$

Dual

$$\max \theta(u)$$
$$u \geq 0$$

where

$$\theta(u) = \min \{ f(x) - ug(x) \mid x \in S \}$$

- Replace objective function $f(x)$ with a lower bound $f(x) - ug(x)$, inequality constraints dropped.
- LP dual is a special case.

Inference dual

Primal

$$\min f(x)$$
$$x \in S$$

Feasible set

Dual

$$\max v$$
$$x \in S \overset{P}{\Rightarrow} v \geq f(x)$$
$$P \in \mathcal{P}$$

Follows using proof P

Family of proofs (inference method)

- Dual is defined relative to an **inference method**.
- Strong duality applies if inference method is **complete**.

Example: LP dual

Primal

$$\min cx$$
$$Ax \geq b$$
$$x \geq 0$$

Dual

$$\max v$$
$$\left\{ \begin{array}{c} Ax \geq b \\ x \geq 0 \end{array} \right\} \overset{P}{\Rightarrow} cx \geq v$$
$$P \in \mathcal{P} \longleftarrow$$

Nonnegative linear combination + domination

Example: LP dual

Primal

$$\min cx$$
$$Ax \geq b$$
$$x \geq 0$$

Dual

$$\max v$$
$$\left\{ \begin{array}{c} Ax \geq b \\ x \geq 0 \end{array} \right\} \overset{P}{\Rightarrow} cx \geq v$$
$$P \in \mathcal{P} \longleftarrow$$ Nonnegative linear combination + domination

$$Ax \geq b \overset{x \geq 0}{\Rightarrow} cx \geq v \quad \text{iff}$$

$uAx \geq ub$ dominates $cx \geq v$

for some $u \geq 0$

$uA \leq c$ and $ub \geq v$

Example: LP dual

Primal

$$\min cx$$
$$Ax \geq b$$
$$x \geq 0$$

Dual

$$\max v \qquad = \max ub$$
$$\left\{ \begin{matrix} Ax \geq b \\ x \geq 0 \end{matrix} \right\} \overset{P}{\Rightarrow} cx \geq v \qquad uA \leq c$$
$$u \geq 0$$

$P \in \mathcal{P} \longleftarrow$ Nonnegative linear combination + domination

$Ax \geq b \overset{x \geq 0}{\Rightarrow} cx \geq v$ iff $uAx \geq ub$ dominates $cx \geq v$

for some $u \geq 0$

$uA \leq c$ and $ub \geq v$

- Inference method is **complete** (assuming feasibility) due to Farkas Lemma.
- So we have **strong duality** (assuming feasibility).

Example: Lagrangean dual

Primal

$$\min f(x)$$
$$g(x) \geq 0$$
$$x \in S$$

Dual

$$\max v$$

$$g(x) \geq b \overset{P}{\Rightarrow} f(x) \geq v$$

$P \in \mathcal{P}$ ⟵ Nonnegative linear combination + domination

Example: Lagrangean dual

Primal	Dual
$\min f(x)$	$\max v$
$g(x) \geq 0$	$g(x) \geq b \overset{P}{\Rightarrow} f(x) \geq v$
$x \in S$	$P \in \mathcal{P}$ ⟵ Nonnegative linear combination + domination

$g(x) \geq 0 \overset{x \in S}{\Rightarrow} f(x) \geq v$ iff $ug(x) \geq 0$ dominates $f(x) - v \geq 0$ for some $u \geq 0$

$ug(x) \leq f(x) - v$ for all $x \in S$

That is, $v \leq f(x) - ug(x)$ for all $x \in S$

Or $v \leq \min_{x \in S} \{f(x) - ug(x)\}$

Example: Lagrangean dual

Primal

$$\min f(x)$$
$$g(x) \geq 0$$
$$x \in S$$

Dual

$$\max v$$
$$g(x) \geq b \overset{P}{\Rightarrow} f(x) \geq v$$
$$P \in \mathcal{P} \longleftarrow$$ Nonnegative linear combination + domination

$$= \max_{u \geq 0} \min_{x \in S} \{f(x) - ug(x)\}$$

$$g(x) \geq 0 \overset{x \in S}{\Rightarrow} f(x) \geq v \quad \text{iff} \quad ug(x) \geq 0 \text{ dominates } f(x) - v \geq 0$$

for some $u \geq 0$

$ug(x) \leq f(x) - v$ for all $x \in S$

That is, $v \leq f(x) - ug(x)$ for all $x \in S$

Or $v \leq \min_{x \in S} \{f(x) - ug(x)\}$

- Inference method is **incomplete**

Example: Surrogate dual

Primal	Dual
$\min f(x)$	$\max v$
$g(x) \geq 0$	$g(x) \geq b \overset{P}{\Rightarrow} f(x) \geq v$
$x \in S$	$P \in \mathcal{P}$ ⟵ Nonnegative linear combination + implication

Example: Surrogate dual

Primal	Dual
$\min f(x)$	$\max v$
$g(x) \geq 0$	$g(x) \geq b \overset{P}{\Rightarrow} f(x) \geq v$
$x \in S$	$P \in \mathcal{P}$ ⟵ Nonnegative linear combination + implication

$g(x) \geq 0 \overset{x \in S}{\Rightarrow} f(x) \geq v$ iff $ug(x) \geq 0$ implies $f(x) \geq v$ for some $u \geq 0$

Any $x \in S$ with $ug(x) \geq 0$ satisfies $f(x) \geq v$

So, $\min \{f(x) \mid ug(x) \leq 0, x \in S\} \geq v$

Example: Surrogate dual

Primal

$$\min f(x)$$
$$g(x) \geq 0$$
$$x \in S$$

Dual

$$\max v = \max_{u \geq 0} \min_{x \in S} \{ f(x) \mid ug(x) \geq 0 \}$$

$$g(x) \geq b \overset{P}{\Rightarrow} f(x) \geq v$$

$P \in \mathcal{P}$ ⟵ Nonnegative linear combination + implication

$$g(x) \geq 0 \overset{x \in S}{\Rightarrow} f(x) \geq v \quad \text{iff} \quad ug(x) \geq 0 \text{ implies } f(x) \geq v \text{ for some } u \geq 0$$

Any $x \in S$ with $ug(x) \geq 0$ satisfies $f(x) \geq v$

So, $\min \{ f(x) \mid ug(x) \leq 0, x \in S \} \geq v$

- Inference method is **incomplete**

Focus on inference duality

Outline:

- Nogood-based search
  - Solution of inference dual provides **nogoods** to guide search.
  - **Benders cuts** and **conflict clauses** in SAT are examples.

Focus on inference duality

Outline:

- Nogood-based search
  - Solution of inference dual provides **nogoods** to guide search.
  - **Benders cuts** and **conflict clauses** in SAT are examples.
- Example: SAT
  - DPL + conflict clauses
  - Partial order dynamic backtracking

Focus on inference duality

Outline:

- Nogood-based search
  - Solution of inference dual provides **nogoods** to guide search.
  - **Benders cuts** and **conflict clauses** in SAT are examples.
- Example: SAT
  - DPL + conflict clauses
  - Partial order dynamic backtracking
- Examples: Integrated problem solving
  - Including more examples of **logic-based Benders**

Nogood-based search

- Nogoods record search experience
  - Each solution examined generates a nogood.
  - Next solution must satisfy current nogood set.
  - Branching search is a special case.

- Nogoods can be derived by solving inference dual of a subproblem.
  - Subproblem can be defined by fixing some variables to current values in the search.

Example: Logic-based Benders

Partition variables x,y and search over values of x

$$\min \ f(x,y)$$
$$(x,y) \in S$$

Subproblem results from fixing x

$$\min \ f(\bar{x},y)$$
$$(\bar{x},y) \in S$$

$\bar{x}$

Hooker 1994
Hooker & Ottosson 2003

Example: Logic-based Benders

Partition variables x,y and search over values of x

$$\min \ f(x,y)$$
$$(x,y) \in S$$

Subproblem results from fixing x

$$\min \ f(\bar{x},y)$$
$$(\bar{x},y) \in S$$

Let proof P be solution of subproblem inference dual for $x = \bar{x}$

Hooker 1994
Hooker & Ottosson 2003

Example: Logic-based Benders

Partition variables x,y and search over values of x

$$\min \ f(x,y)$$
$$(x,y) \in S$$

Subproblem results from fixing x

$$\min \ f(\bar{x},y)$$
$$(\bar{x},y) \in S$$

Let proof P be solution of subproblem inference dual for $x = \bar{x}$

Then P derives a lower bound $B(P,\bar{x})$

$\bar{x}$

Hooker 1994
Hooker & Ottosson 2003

Example: Logic-based Benders

Partition variables x,y and search over values of x

$$\min \ f(x,y)$$
$$(x,y) \in S$$

Subproblem results from fixing x

$$\min \ f(\bar{x},y)$$
$$(\bar{x},y) \in S$$

Let proof P be solution of subproblem inference dual for $x = \bar{x}$

Then P derives a lower bound $B(P,\bar{x})$

The same proof P provides a lower bound $B(P,x)$ for any x

$\bar{x}$

Hooker 1994
Hooker & Ottosson 2003

Example: Logic-based Benders

Partition variables x,y and search over values of x

$$\min \ f(x,y)$$
$$(x,y) \in S$$

Subproblem results from fixing x

$$\min \ f(\bar{x},y)$$
$$(\bar{x},y) \in S$$

Let proof P be solution of subproblem inference dual for $x = \bar{x}$

Then P derives a lower bound $B(P,\bar{x})$

The same proof P provides a lower bound $B(P,x)$ for any x

Add the **Benders cut** $v \geq B(P,x)$ to the **master problem** $\min v$

$\bar{x}$ Benders cuts

Hooker 1994
Hooker & Ottosson 2003

Example: Logic-based Benders

Partition variables x,y and search over values of x

$$\min \ f(x,y)$$
$$(x,y) \in S$$

Subproblem results from fixing x

$$\min \ f(\bar{x},y)$$
$$(\bar{x},y) \in S$$

Let proof P be solution of subproblem inference dual for $x = \bar{x}$

Then P derives a lower bound $B(P,\bar{x})$

The same proof P provides a lower bound $B(P,x)$ for any x

Add the **Benders cut** $v \geq B(P,x)$ to the **master problem** $\min v$

Solve master problem for next $\bar{x}$ Benders cuts

Hooker 1994
Hooker & Ottosson 2003

Classical Benders

Partition variables x,y and search over values of x

$$\min \; f(x) + cy$$
$$g(x) + Ay \geq b$$
$$x \in D_x, \;\; y \geq 0$$

Subproblem results from fixing x

$$\min \; f(\bar{x}) + cy$$
$$Ay \geq b - g(\bar{x}) \quad (u)$$
$$y \geq 0$$

Classical Benders

Partition variables x,y and search over values of x

$$\min \ f(x) + cy$$
$$g(x) + Ay \geq b$$
$$x \in D_x, \ y \geq 0$$

Subproblem results from fixing x

$$\min \ f(\bar{x}) + cy$$
$$Ay \geq b - g(\bar{x}) \quad (u)$$
$$y \geq 0$$

Let u be a dual solution of subproblem for $x = \bar{x}$

Classical Benders

Partition variables x,y and search over values of x

$$\min \; f(x) + cy$$
$$g(x) + Ay \geq b$$
$$x \in D_x, \; y \geq 0$$

Subproblem results from fixing x

$$\min \; f(\bar{x}) + cy$$
$$Ay \geq b - g(\bar{x}) \quad (u)$$
$$y \geq 0$$

Let u be a dual solution of subproblem for $x = \bar{x}$

Then u is proof of bound $B(u, \bar{x}) = f(\bar{x}) + u(b - g(\bar{x}))$

$\bar{x}$

Classical Benders

Partition variables x,y and search over values of x

$$\min \ f(x) + cy$$
$$g(x) + Ay \geq b$$
$$x \in D_x, \ y \geq 0$$

Subproblem results from fixing x

$$\min \ f(\bar{x}) + cy$$
$$Ay \geq b - g(\bar{x}) \quad (u)$$
$$y \geq 0$$

Let u be a dual solution of subproblem for $x = \bar{x}$

Then u is proof of bound $B(u,\bar{x}) = f(\bar{x}) + u(b - g(\bar{x}))$

Also u is proof of bound $B(u,x) = f(x) + u(b - g(x))$ for any x

Classical Benders

Partition variables x,y and search over values of x

$$\min \ f(x) + cy$$
$$g(x) + Ay \geq b$$
$$x \in D_x, \ y \geq 0$$

Subproblem results from fixing x

$$\min \ f(\bar{x}) + cy$$
$$Ay \geq b - g(\bar{x}) \quad (u)$$
$$y \geq 0$$

Let u be a dual solution of subproblem for $x = \bar{x}$

Then u is proof of bound $B(u,\bar{x}) = f(\bar{x}) + u(b - g(\bar{x}))$

Also u is proof of bound $B(u,x) = f(x) + u(b - g(x))$ for any x

Add Benders cut $v \geq = f(x) + u(b - g(x))$ to **master problem**:

$\min v$

Benders cuts

Classical Benders

Partition variables x,y and search over values of x

$$\min \ f(x) + cy$$
$$g(x) + Ay \geq b$$
$$x \in D_x, \ y \geq 0$$

Subproblem results from fixing x

$$\min \ f(\bar{x}) + cy$$
$$Ay \geq b - g(\bar{x}) \quad (u)$$
$$y \geq 0$$

Let u be a dual solution of subproblem for $x = \bar{x}$

Then u is proof of bound $B(u,\bar{x}) = f(\bar{x}) + u(b - g(\bar{x}))$

Also u is proof of bound $B(u,x) = f(x) + u(b - g(x))$ for any x

Add Benders cut $v \geq = f(x) + u(b - g(x))$ to **master problem**:

$$\min v$$

Benders cuts

Solve master problem for next $\bar{x}$

Example: SAT

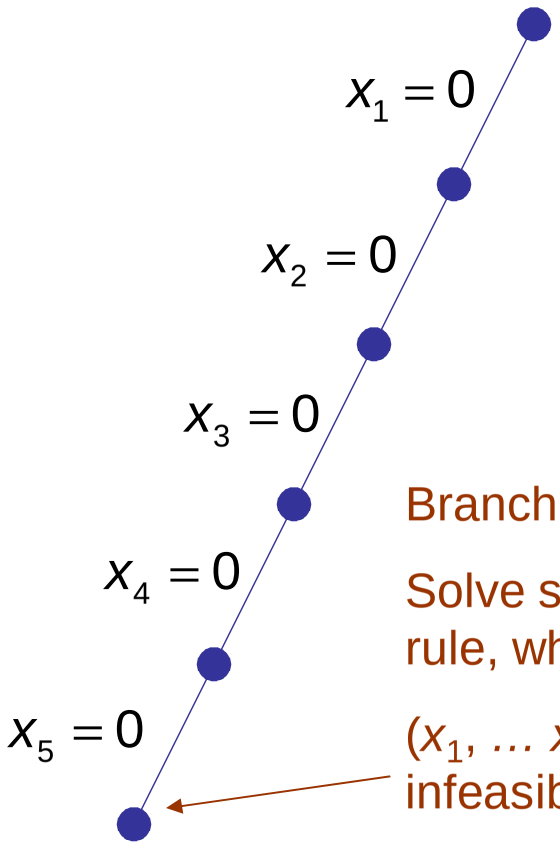
- Solve SAT by chronological backtracking + unit clause rule = DPL
- To get nogood, solve **inference dual** at current node.
  - Solve dual with unit clause rule
- Process nogood set (master problem) with **parallel resolution**
  - Nogood set is a **relaxation** of the problem.

Example: SAT

Find a satisfying solution.

$$x_1 \vee x_5 \vee x_6$$
$$x_1 \vee x_5 \vee \overline{x}_6$$
$$x_2 \vee \overline{x}_5 \vee x_6$$
$$x_2 \vee \overline{x}_5 \vee \overline{x}_6$$
$$\overline{x}_1 \vee x_3 \vee x_4$$
$$\overline{x}_2 \vee x_3 \vee x_4$$
$$\overline{x}_1 \vee \overline{x}_3$$
$$\overline{x}_1 \vee \overline{x}_4$$
$$\overline{x}_2 \vee \overline{x}_3$$
$$\overline{x}_2 \vee \overline{x}_4$$

DPL with chronological backtracking

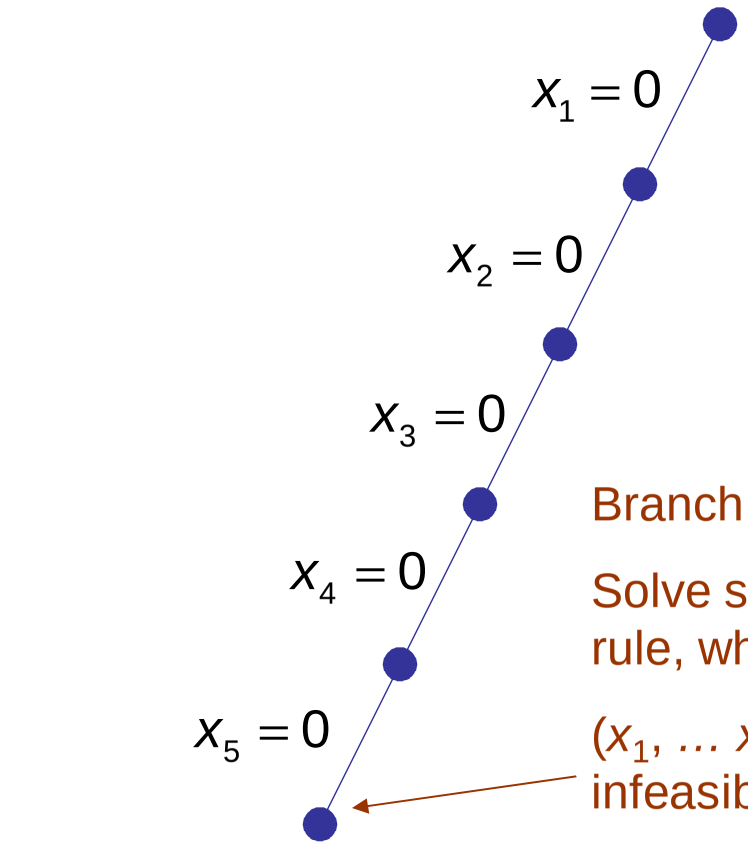


Branch to here.

Solve subproblem with unit clause rule, which proves infeasibility.

(x_1, … x_5) = (0, …, 0) creates the infeasibility.

DPL with chronological backtracking



Branch to here.

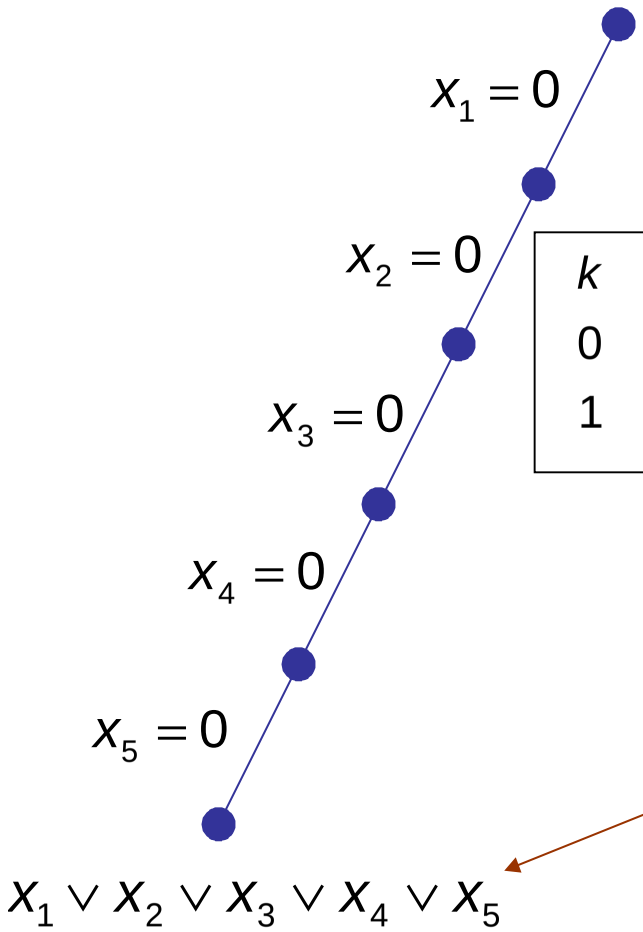
Solve subproblem with unit clause rule, which proves infeasibility.

(x_1, … x_5) = (0, …, 0) creates the infeasibility.

$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$

Generate nogood.

DPL with chronological backtracking

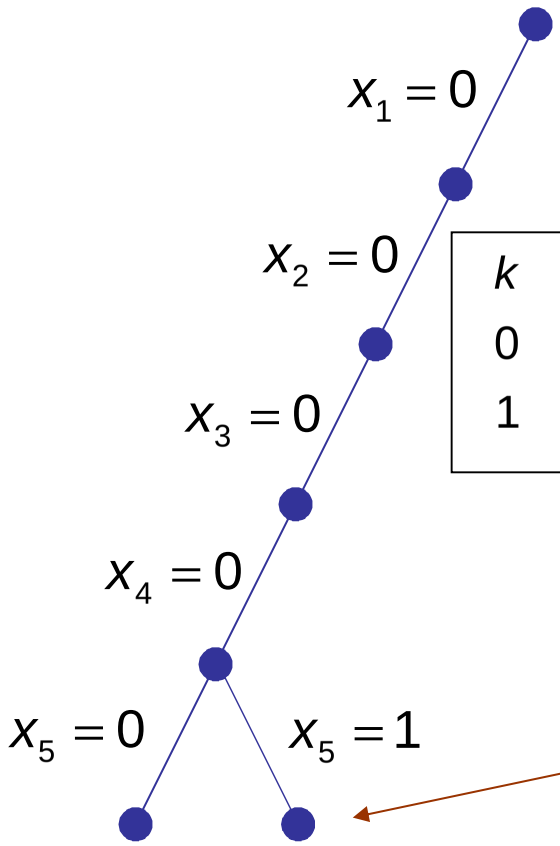


Master problem

k	Relaxation R_k	Solution of R_k	Nogoods
0		$(0,0,0,0,0,\cdot)$	$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$
1	$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$		

Conflict clause appears as nogood induced by solution of R_k.

DPL with chronological backtracking

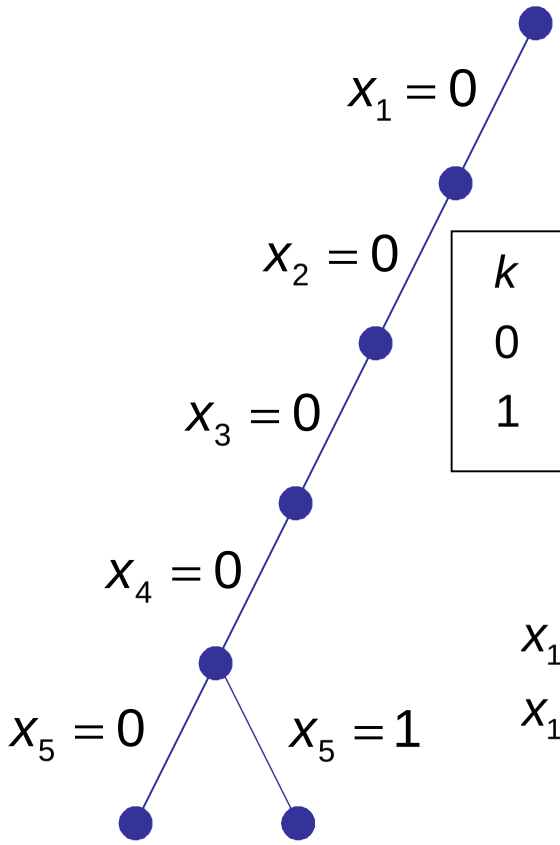


Master problem

k	Relaxation R_k	Solution of R_k	Nogoods
0		$(0,0,0,0,0,\cdot)$	$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$
1	$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$	$(0,0,0,0,1,\cdot)$	$x_1 \vee x_2 \vee x_3 \vee x_4 \vee \overline{x}_5$

Go to solution that solves relaxation, with priority to 0

DPL with chronological backtracking



Master problem

k	Relaxation R_k	Solution of R_k	Nogoods
0		$(0,0,0,0,0,\cdot)$	$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$
1	$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$	$(0,0,0,0,1,\cdot)$	$x_1 \vee x_2 \vee x_3 \vee x_4 \vee \overline{x}_5$

$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$

$x_1 \vee x_2 \vee x_3 \vee x_4 \vee \overline{x}_5$

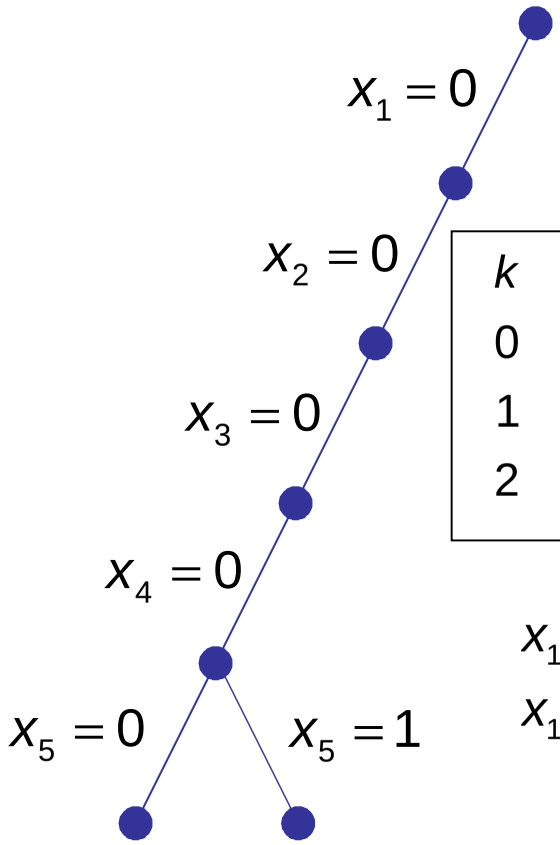
Process nogood set with **parallel resolution**

$x_1 \vee x_2 \vee x_3 \vee x_4$ **parallel-absorbs**

$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$

$x_1 \vee x_2 \vee x_3 \vee x_4 \vee \overline{x}_5$

DPL with chronological backtracking



Master problem

k	Relaxation R_k	Solution of R_k	Nogoods
0		$(0,0,0,0,0,\cdot)$	$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$
1	$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$	$(0,0,0,0,1,\cdot)$	$x_1 \vee x_2 \vee x_3 \vee x_4 \vee \overline{x}_5$
2	$x_1 \vee x_2 \vee x_3 \vee x_4$		

$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$

$x_1 \vee x_2 \vee x_3 \vee x_4 \vee \overline{x}_5$

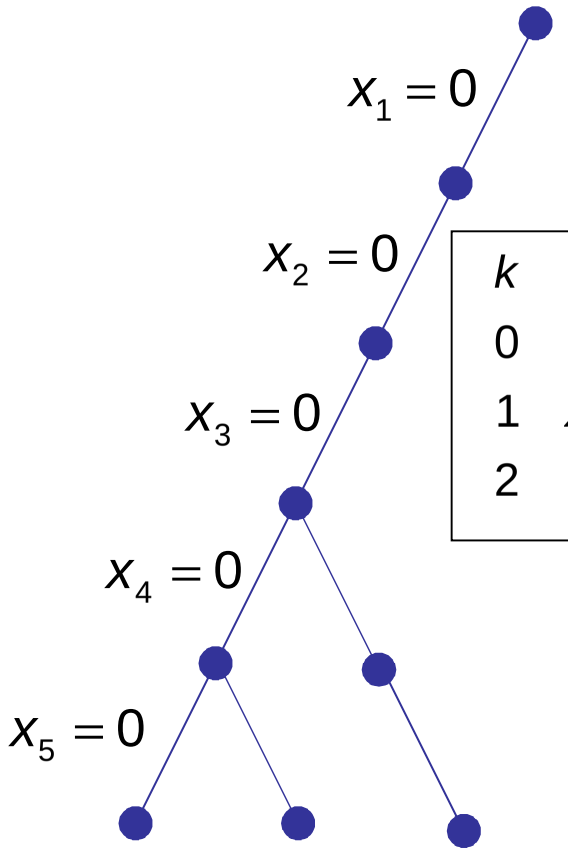
Process nogood set with **parallel resolution**

$x_1 \vee x_2 \vee x_3 \vee x_4$ **parallel-absorbs**

$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$

$x_1 \vee x_2 \vee x_3 \vee x_4 \vee \overline{x}_5$

DPL with chronological backtracking



Master problem

k	Relaxation R_k	Solution of R_k	Nogoods
0		$(0,0,0,0,0,\cdot)$	$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$
1	$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$	$(0,0,0,0,1,\cdot)$	$x_1 \vee x_2 \vee x_3 \vee x_4 \vee \overline{x}_5$
2	$x_1 \vee x_2 \vee x_3 \vee x_4$	$(0,0,0,1,0,\cdot)$	

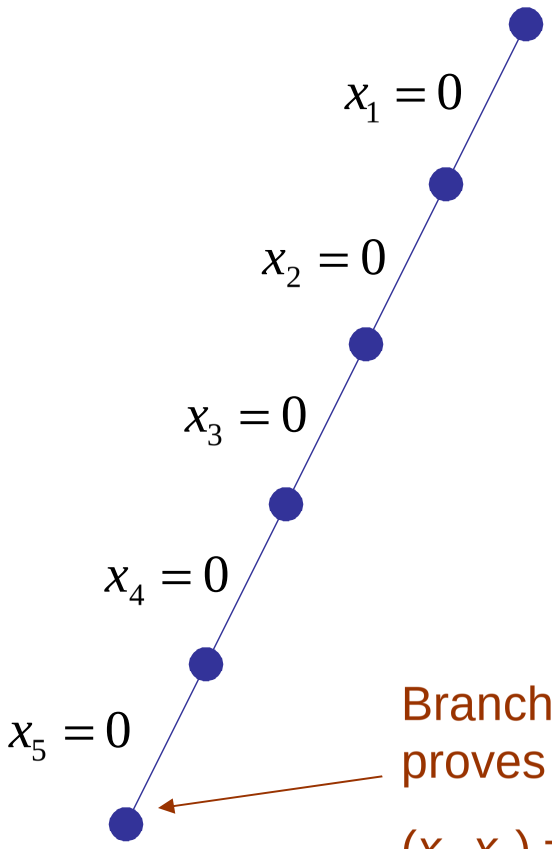
Solve relaxation again, continue.

So backtracking is nogood-based search with parallel resolution

Example: SAT + conflict clauses

- Use stronger nogoods = conflict clauses.
  - Nogoods rule out only branches that play a role in unit clause refutation.

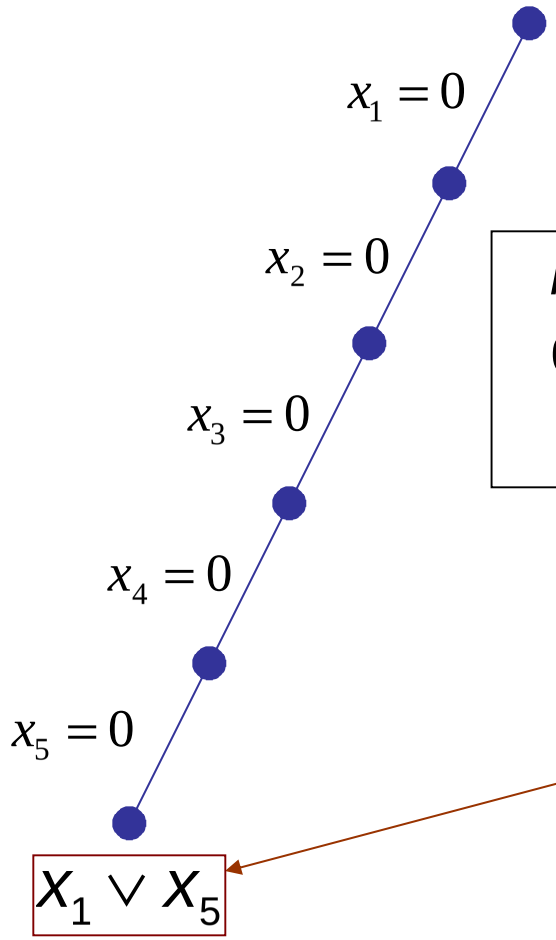
DPL with conflict clauses



Branch to here. Unit clause rule proves infeasibility.

(x_1,x_5) = (0,0) is only premise of unit clause proof.

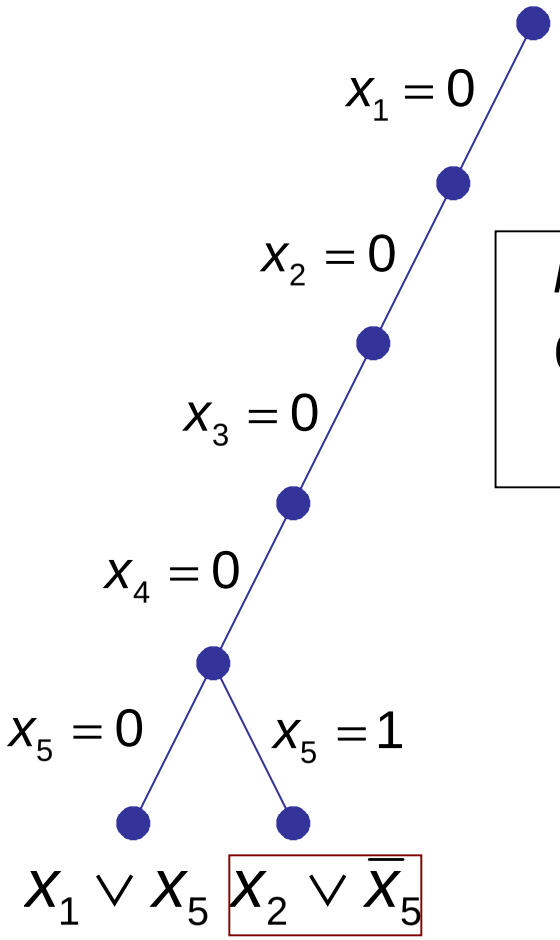
DPL with conflict clauses



k	Relaxation R_k	Solution of R_k	Nogoods
0		$(0,0,0,0,0,\cdot)$	$x_1 \vee x_5$
1	$x_1 \vee x_5$		

Conflict clause appears as nogood induced by solution of R_k.

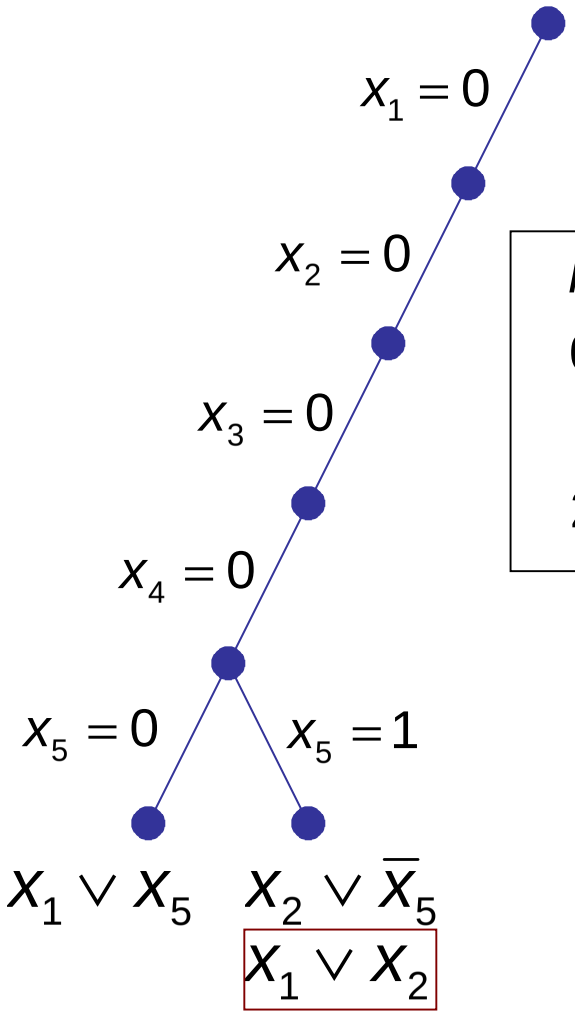
DPL with conflict clauses



Master problem

k	Relaxation R_k	Solution of R_k	Nogoods
0		$(0,0,0,0,0,\cdot)$	$x_1 \vee x_5$
1	$x_1 \vee x_5$	$(0,0,0,0,1,\cdot)$	$x_2 \vee \overline{x}_5$

DPL with conflict clauses



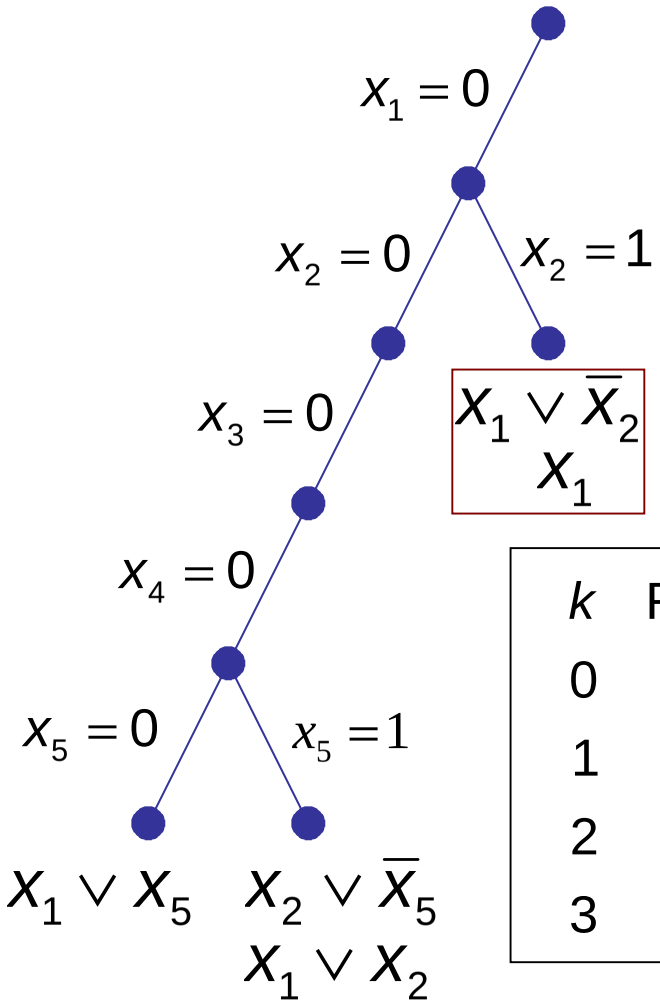
Master problem

k	Relaxation R_k	Solution of R_k	Nogoods
0		$(0,0,0,0,0,\cdot)$	$x_1 \vee x_5$
1	$x_1 \vee x_5$	$(0,0,0,0,1,\cdot)$	$x_2 \vee \bar{x}_5$
2	$x_1 \vee x_2$		

$x_1 \vee x_5$
$x_2 \vee \bar{x}_5$ parallel-resolve to yield $x_1 \vee x_2$

$x_1 \vee x_2$ parallel-absorbs $x_1 \vee x_5$, $x_2 \vee \bar{x}_5$

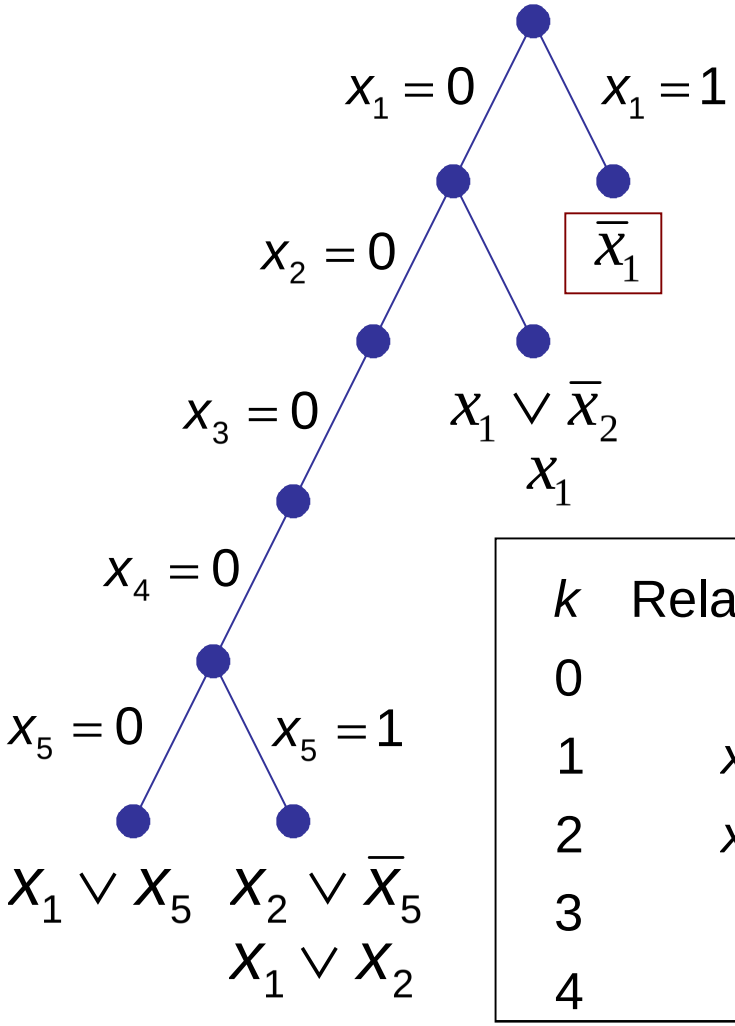
DPL with conflict clauses



k	Relaxation R_k	Solution of R_k	Nogoods
0		$(0,0,0,0,0,\cdot)$	$x_1 \vee x_5$
1	$x_1 \vee x_5$	$(0,0,0,0,1,\cdot)$	$x_2 \vee \overline{x}_5$
2	$x_1 \vee x_2$	$(0,1,\cdot,\cdot,\cdot,\cdot)$	$x_1 \vee \overline{x}_2$
3	x_1		

$x_1 \vee x_2$
$x_1 \vee \overline{x}_2$ parallel-resolve to yield x_1

DPL with conflict clauses



k	Relaxation R_k	Solution of R_k	Nogoods
0		$(0,0,0,0,0,\cdot)$	$x_1 \vee x_5$
1	$x_1 \vee x_5$	$(0,0,0,0,1,\cdot)$	$x_2 \vee \overline{x}_5$
2	$x_1 \vee x_2$	$(0,1,\cdot,\cdot,\cdot,\cdot)$	$x_1 \vee \overline{x}_2$
3	x_1	$(1,\cdot,\cdot,\cdot,\cdot,\cdot)$	$\overline{x}_1$
4	$\emptyset$		

Search terminates

SAT + partial order dynamic backtracking

- Forget about the branching tree
  - Let nogoods direct the search in a more general way.
  - Similar method used in **recent SAT solvers**

- Solve relaxation by selecting a solution that **conforms** to nogoods.
  - Conform = takes opposite sign than in nogoods.
  - More freedom than in branching.

Ginsberg & McAllister 1993, 1994

Partial Order Dynamic Backtracking

k	Relaxation R_k	Solution of R_k	Nogoods
0		$(0,0,0,0,0,\cdot)$	$x_5 \vee x_1$
1	$x_1 \vee x_5$		

Arbitrarily choose one variable to be last

Partial Order Dynamic Backtracking

k	Relaxation R_k	Solution of R_k	Nogoods
0		$(0,0,0,0,0,\cdot)$	$x_5 \vee x_1$
1	$x_1 \vee x_5$		

Other variables are penultimate

Arbitrarily choose one variable to be last

Partial Order Dynamic Backtracking

k	Relaxation R_k	Solution of R_k	Nogoods
0		$(0,0,0,0,0,\cdot)$	$x_5 \vee x_1$
1	$x_1 \vee x_5$	$(1,\cdot,\cdot,\cdot,0,\cdot)$	
2			

Since x_5 is penultimate in at least one nogood, it must conform to nogoods.

It must take value opposite its sign in the nogoods.

x_5 will have the same sign in all nogoods where it is penultimate.

This allows more freedom than chronological backtracking.

Partial Order Dynamic Backtracking

k	Relaxation R_k	Solution of R_k	Nogoods
0		$(0,0,0,0,0,\cdot)$	$x_5 \vee x_1$
1	$x_1 \vee x_5$	$(1,\cdot,\cdot,\cdot,0,\cdot)$	$x_5 \vee \overline{x}_1$
2			

Choice of last variable is arbitrary but must be consistent with partial order implied by previous choices.

Since x_5 is penultimate in at least one nogood, it must conform to nogoods.

It must take value opposite its sign in the nogoods.

x_5 will have the same sign in all nogoods where it is penultimate.

This allows more freedom than chronological backtracking.

Partial Order Dynamic Backtracking

k	Relaxation R_k	Solution of R_k	Nogoods
0		$(0,0,0,0,0,\cdot)$	$x_5 \vee x_1$
1	$x_1 \vee x_5$	$(1,\cdot,\cdot,\cdot,0,\cdot)$	$x_5 \vee \overline{x}_1$
2	x_5		

Choice of last variable is arbitrary but must be consistent with partial order implied by previous choices.

$x_5 \vee x_1$
$x_5 \vee \overline{x}_1$

Parallel-resolve to yield x_5

Since x_5 is penultimate in at least one nogood, it must conform to nogoods.

It must take value opposite its sign in the nogoods.

x_5 will have the same sign in all nogoods where it is penultimate.

This allows more freedom than chronological backtracking.

Partial Order Dynamic Backtracking

k	Relaxation R_k	Solution of R_k	Nogoods
0		$(0,0,0,0,0,\cdot)$	$x_5 \vee x_1$
1	$x_1 \vee x_5$	$(1,\cdot,\cdot,\cdot,0,\cdot)$	$x_5 \vee \overline{x}_1$
2	x_5	$(\cdot,0,\cdot,\cdot,1,\cdot)$	$\overline{x}_5 \vee x_2$
3	$\{x_5, \overline{x}_5 \vee x_2\}$		

x_5 does not parallel-resolve with $\overline{x}_5 \vee x_2$

because x_5 is not last in both clauses

Partial Order Dynamic Backtracking

k	Relaxation R_k	Solution of R_k	Nogoods
0		$(0,0,0,0,0,\cdot)$	$x_5 \vee x_1$
1	$x_1 \vee x_5$	$(1,\cdot,\cdot,\cdot,0,\cdot)$	$x_5 \vee \bar{x}_1$
2	x_5	$(\cdot,0,\cdot,\cdot,1,\cdot)$	$\bar{x}_5 \vee x_2$
3	$\{x_5, \bar{x}_5 \vee x_2\}$	$(\cdot,1,\cdot,\cdot,1,\cdot)$	$\bar{x}_2$
4	$\varnothing$		

Search terminates

Must conform

Examples of integrated solving

- Piecewise linear optimization.
  - Illustrates modeling with metaconstraint
- Planning & scheduling - Logic-based Benders
  - Planning and disjunctive scheduling
  - Planning and cumulative scheduling
  - Single-facility scheduling
- Truss structure design - Global optimization.

- Solved by SIMPL, a prototype integrated solver.

Aron, Hooker, & Yunes 2004, 2010

Piecewise linear optimization

Maximize a piecewise linear separable function subject to budget constraint.

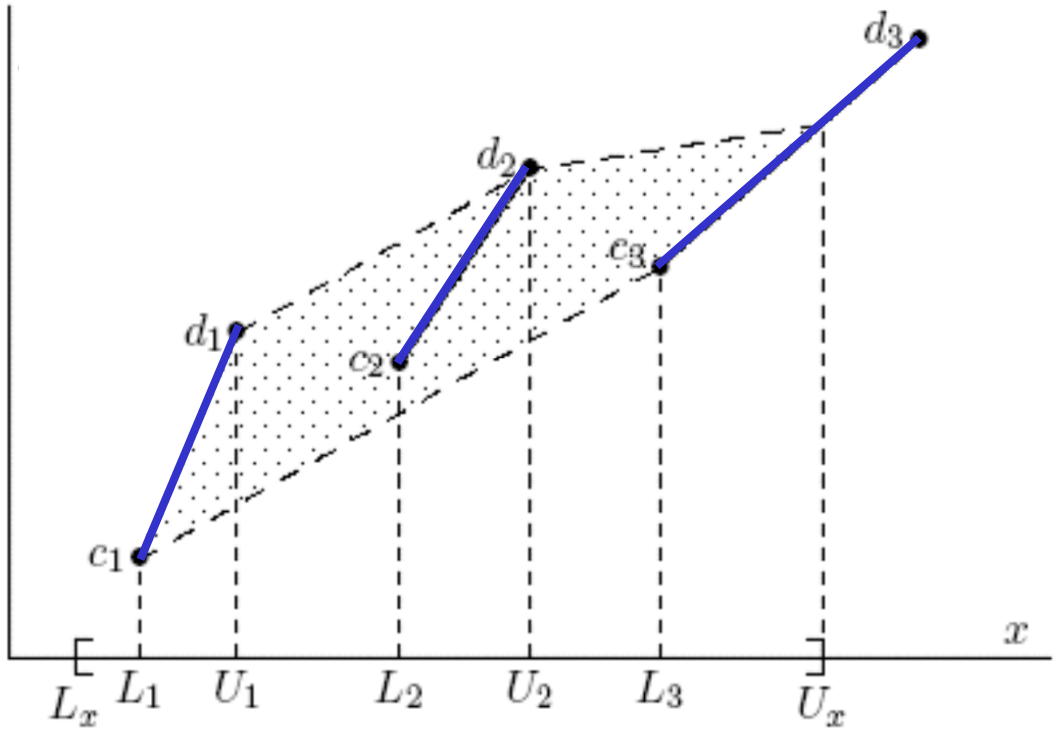
$$\max \sum_i f_i(x_i)$$

$$\sum_i x_i \le C$$

Each f_i is a piecewise linear semicontinuous function

Piecewise linear optimization

Semicontinuous piecewise linear function *f*(*x*)



Piecewise linear optimization

Integrated approach

- *Search:* **Branch** on variables
- *Relaxation:* Use a specialized **convex hull relaxation** for piecewise linear functions (no need for 0-1 model or SOS2).
- *Inference:* **Bounds propagation**.

Ottosson, Thorsteinsson & Hooker 1999

Piecewise linear optimization

Integrated model

$$\max \sum_i u_i$$

$$\sum_i x_i \le C$$

$$\text{piecewise}\left(x_i, u_i, L_i, U_i, c_i, d_i\right), \quad \text{all } i$$

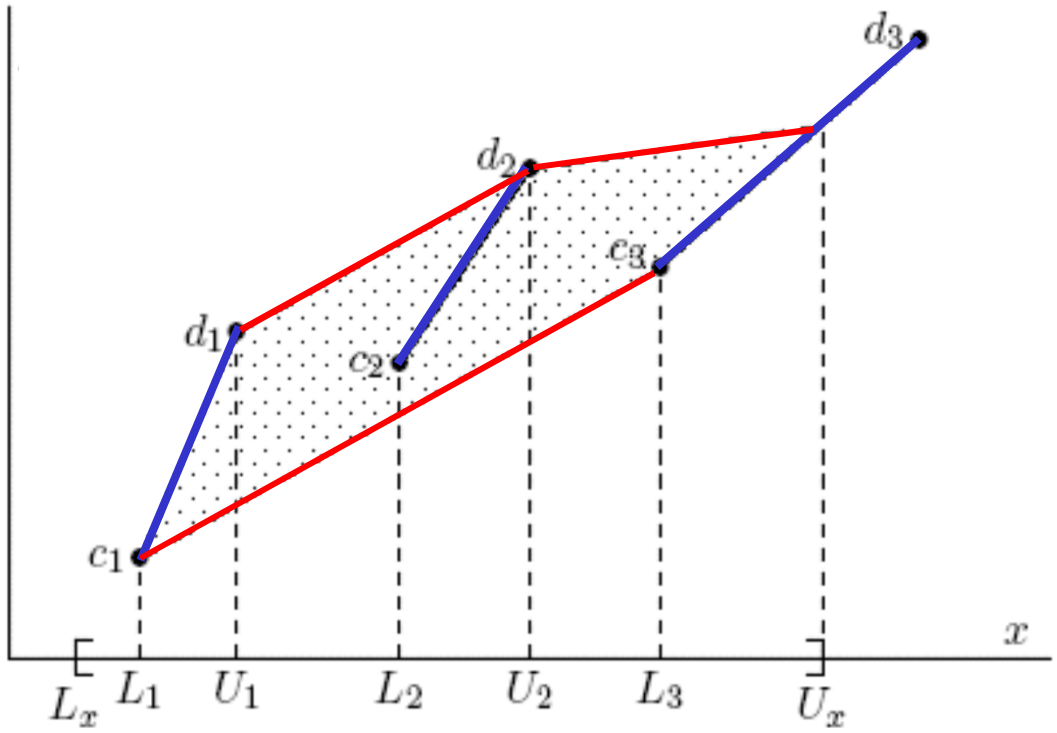
Metaconstraint

(global constraint in CP)

Piecewise linear optimization

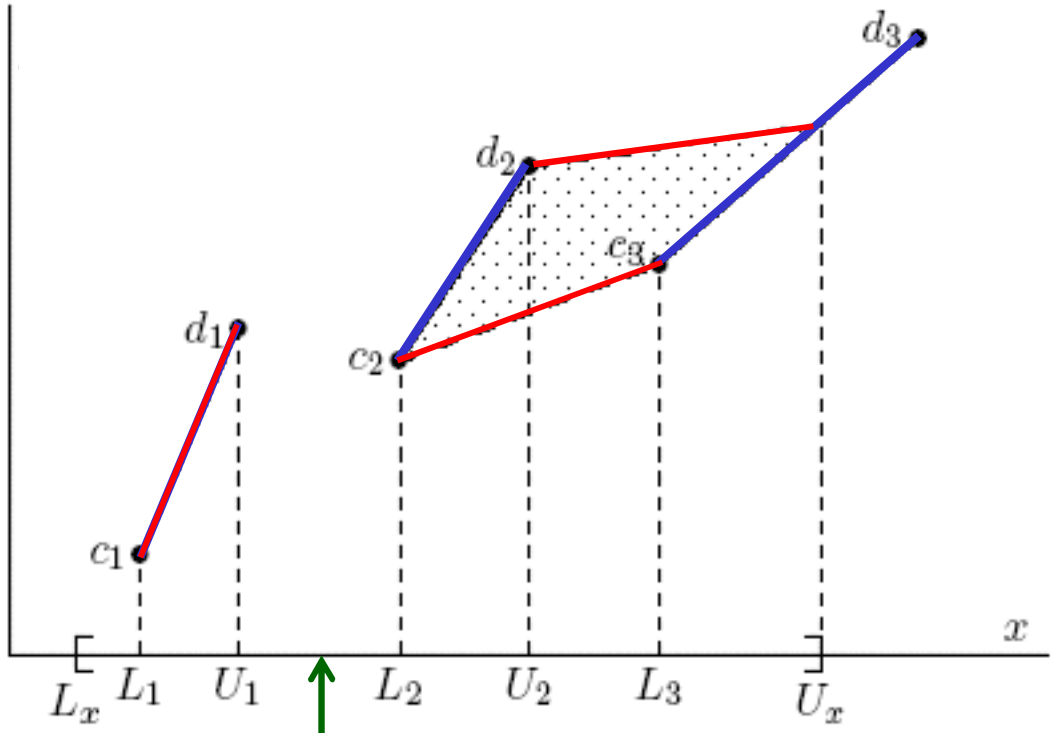
Semicontinuous piecewise linear function $f(x)$

Tight linear relaxation after propagation



Piecewise linear optimization

Semicontinuous piecewise linear function *f*(*x*)



Tighter relaxation after branching

Value of x in solution of current linear relaxation

Piecewise linear optimization

SIMPL model

```
01. OBJECTIVE
02.   maximize sum i of u[i]
03. CONSTRAINTS
04.   capacity means {
05.     sum i of x[i] <= C
06.     relaxation = { lp, cp } }
07.   piecewisectr means {
08.     piecewise(x[i],u[i],L[i],U[i],c[i],d[i]) forall i
09.     relaxation = { lp, cp } }
10. SEARCH
11.   type = { bb:bestdive }
12.   branching = { piecewisectr:most }
```

Piecewise linear optimization

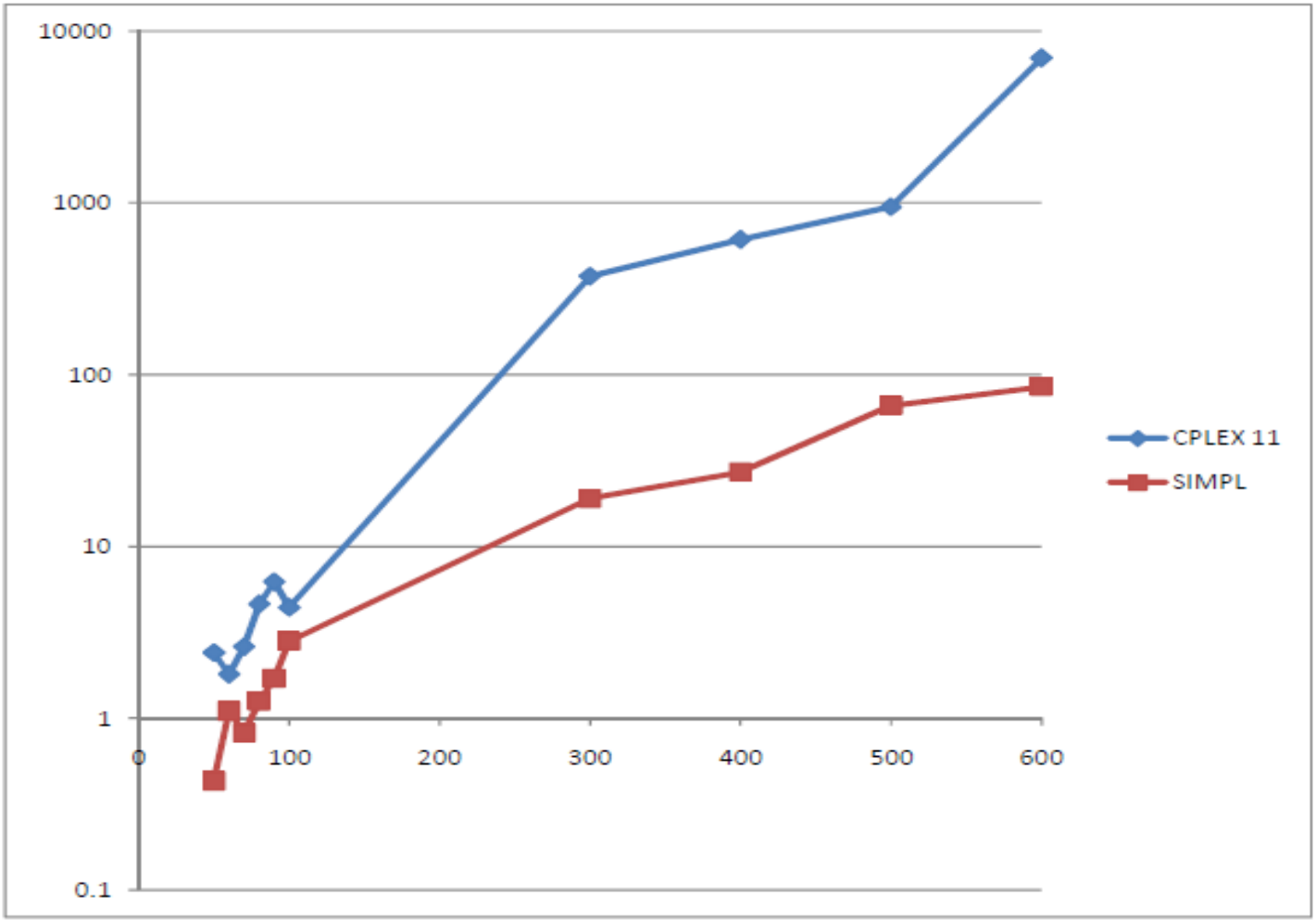
SIMPL model

```
01. OBJECTIVE
02.   maximize sum i of u[i]
03. CONSTRAINTS
04.   capacity means {
05.     sum i of x[i] <= C
06.     relaxation = { lp, cp } }
07.   piecewisectr means {
08.     piecewise(x[i],u[i],L[i],U[i],c[i],d[i]) forall i
09.     relaxation = { lp, cp } }
10. SEARCH
11.   type = { bb:bestdive }
12.   branching = { piecewisectr:most }
```

Piecewise linear metaconstraint.

Piecewise linear optimization

Computation time (sec) vs. number of jobs



Planning and disjunctive scheduling

- Assign jobs to machines, and schedule the machines assigned to each machine within time windows.
- The objective is to minimize **processing cost**, **makespan**, or **total tardiness**.

Planning and disjunctive scheduling

Job Data

Job j	*Release time* r_j	*Dead-line* d_j	*Processing time* p_{Aj}	p_{Bj}
1	0	9	1	5
2	0	9	3	6
3	2	7	3	7
4	2	9	4	6
5	4	7	2	5

Machine A

Machine B

Example

Assign 5 jobs to 2 machines.

Schedule jobs assigned to each machine without overlap (disjunctive scheduling)

Planning and disjunctive scheduling

Integrated model

$$\min \sum_j c_{x_j j}$$

$$r_j \le s_j \le d_j - p_{x_j j}, \text{ all } j$$

$$\text{noOverlap}\left((s_j \mid x_j = i), (p_{ij} \mid x_j = i)\right), \text{ all } i$$

Start time of job *j*

Time windows

Jobs cannot overlap

Machine assigned to job *j*

Here we minimize processing cost.

Planning and disjunctive scheduling

Integrated approach

• *Search:* Enumerate **subproblems** (defined by assigning jobs to machines)

• *Relaxation:* Enumerate **master problems** (which assign jobs to machines)

• *Inference:* Generate **nogoods** (logic-based Benders cuts), which are added to master problem.

Planning and disjunctive scheduling

Integrated approach

- Assign the jobs in the **master problem**, to be solved by **MILP**.
- Schedule the jobs in the **subproblem**, to be solved by **CP**.

Jain & Grossmann 2001

Planning and disjunctive scheduling

Integrated approach

- Assign the jobs in the **master problem**, to be solved by **MILP**.
- Schedule the jobs in the **subproblem**, to be solved by **CP**.

The subproblem decouples into a separate scheduling problem on each machine.

In this problem, the subproblem is a feasibility problem.

Jain & Grossmann 2001

Planning and disjunctive scheduling

Integrated approach

- Assign the jobs in the **master problem**, to be solved by **MILP**.
- Schedule the jobs in the **subproblem**, to be solved by **CP**.

The subproblem decouples into a separate scheduling problem on each machine.

In this problem, the subproblem is a feasibility problem.

- Solve **inference dual** of subproblem to generate **nogoods** (logic-based Benders cuts), which are added to master problem.

Planning and disjunctive scheduling

Integrated model

$$\min \sum_j c_{x_j j}$$

Start time of job *j*

$$r_j \le s_j \le d_j - p_{x_j j}, \text{ all } j$$

Time windows

$$\text{noOverlap}\left((s_j \,|\, x_j = i), (p_{ij} \,|\, x_j = i)\right), \text{ all } i$$

Jobs cannot overlap

For a fixed assignment $\bar{x}$ the subproblem on each machine *i* is

$$r_j \le s_j \le d_j - p_{\bar{x}_j j}, \text{ all } j \text{ with } \bar{x}_j = i$$

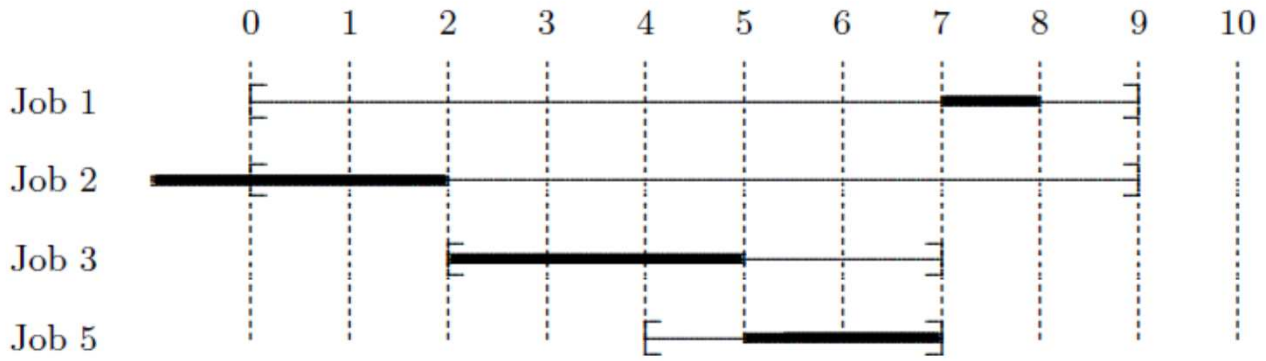
$$\text{noOverlap}\left((s_j \,|\, \bar{x}_j = i), (p_{ij} \,|\, \bar{x}_j = i)\right)$$

Planning and disjunctive scheduling

Logic-based Benders approach

Suppose we assign jobs 1,2,3,5 to machine A in iteration *k*.

We can prove that there is no feasible schedule.



Edge finding derives infeasibility by reasoning only with jobs 2,3,5.
So these jobs alone create infeasibility.

So we have a Benders cut $\neg(x_2 = x_3 = x_5 = A)$

Planning and disjunctive scheduling

Logic-based Benders approach

We want the master problem to be an MILP, which is good for assignment problems.

So we write the Benders cut $\neg(x_2 = x_3 = x_5 = A)$

Using 0-1 variables: $x_{A2} + x_{A3} + \boxed{x_{A5}} \leq 2$

= 1 if job 5 is assigned to machine A

Planning and disjunctive scheduling

The master problem is a **relaxation**, formulated as an MILP:

$$\min \sum_{ij} c_{ij} x_{ij}$$

$$x_{A2} + x_{A3} + x_{A5} \leq 2$$

relaxation of subproblem

$$x_{ij} \in \{0,1\}$$

Planning and disjunctive scheduling

- Keys to success:
  - **Strengthen** Benders cuts by solving subproblems for a neighborhood of current master solution.
  - Use **relaxation** of subproblem in the master.

Planning and disjunctive scheduling

SIMPL model

```
01. OBJECTIVE
02.   min sum i,j of c[i][j] * x[i][j];
03. CONSTRAINTS
04.   assign means {
05.     sum i of x[i][j] = 1 forall j;
06.     relaxation = { ip:master } }
07.   xy means {
08.     x[i][j] = 1 <=> y[j] = i forall i, j;
09.     relaxation = { cp } }
10.   tbounds means {
11.     r[j] <= t[j] forall j;
12.     t[j] <= d[j] - p[y[j]][j] forall j;
13.     relaxation = { ip:master, cp } }
14.   machinecap means {
15.     cumulative({ t[j], p[i][j], 1 } forall j | x[i][j] = 1, 1) forall i;
16.     relaxation = { cp:subproblem, ip:master }
17.     inference = { feasibility } }
18. SEARCH
19.   type = { benders }
```

Planning and disjunctive scheduling

Yunes, Aron & Hooker 2010

Computational results – Long processing times

Jobs	Machines	MILP (CPLEX 11) Nodes	MILP (CPLEX 11) Sec.	SIMPL Benders Iter.	SIMPL Benders Cuts	SIMPL Benders Sec.
3	2	1	0.00	2	1	0.00
7	3	1	0.00	13	16	0.12
12	3	3,351	6.6	26	35	0.73
15	5	2,779	8.8	20	29	0.83
20	5	33,321	882	13	82	5.4
22	5	352,309	10,563	69	98	9.6

SIMPL results are similar to original hand-coded results.

Planning and disjunctive scheduling

Yunes, Aron & Hooker 2010

Computational results – Short processing times

Jobs	Machines	MILP (CPLEX 11) Nodes	MILP (CPLEX 11) Sec.	SIMPL Benders Iter.	SIMPL Benders Cuts	SIMPL Benders Sec.
3	2	1	0.01	1	0	0.00
7	3	1	0.02	1	0	0.00
12	3	499	0.98	1	0	0.01
15	5	529	2.6	2	1	0.06
20	5	250,047	369	6	5	0.28
22	5	> 27.5 mil.	> 48 hr	9	12	0.42
25	5	> 5.4 mil.	> 19 hr*	17	21	1.09

*out of memory

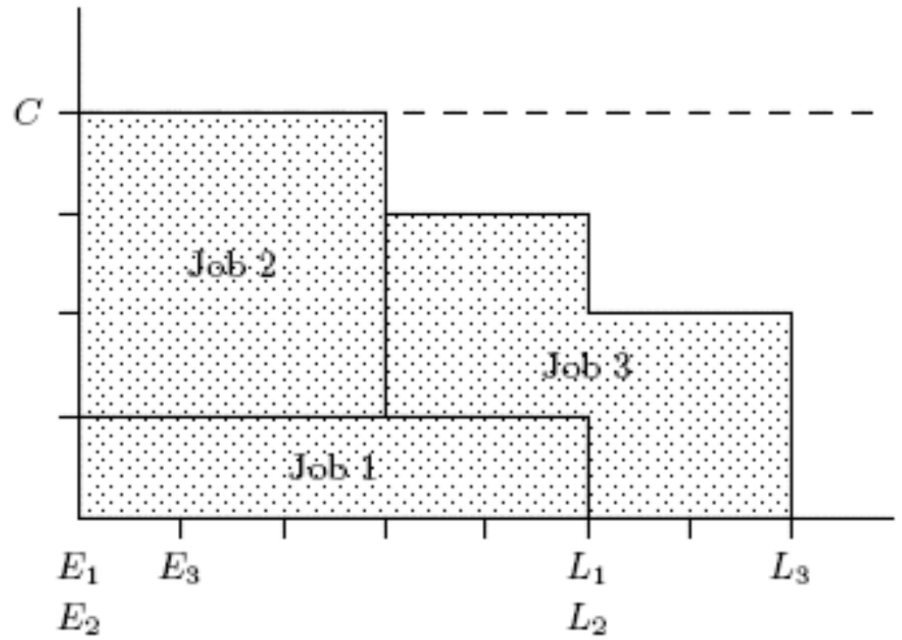
Planning and cumulative scheduling

Consider a cumulative scheduling constraint:

$$\text{cumulative}\big((s_1,s_2,s_3),(p_1,p_2,p_3),(c_1,c_2,c_3),C\big)$$

j	p_j	c_j	E_j	L_j
1	5	1	0	5
2	3	3	0	5
3	4	2	1	7

A feasible solution:

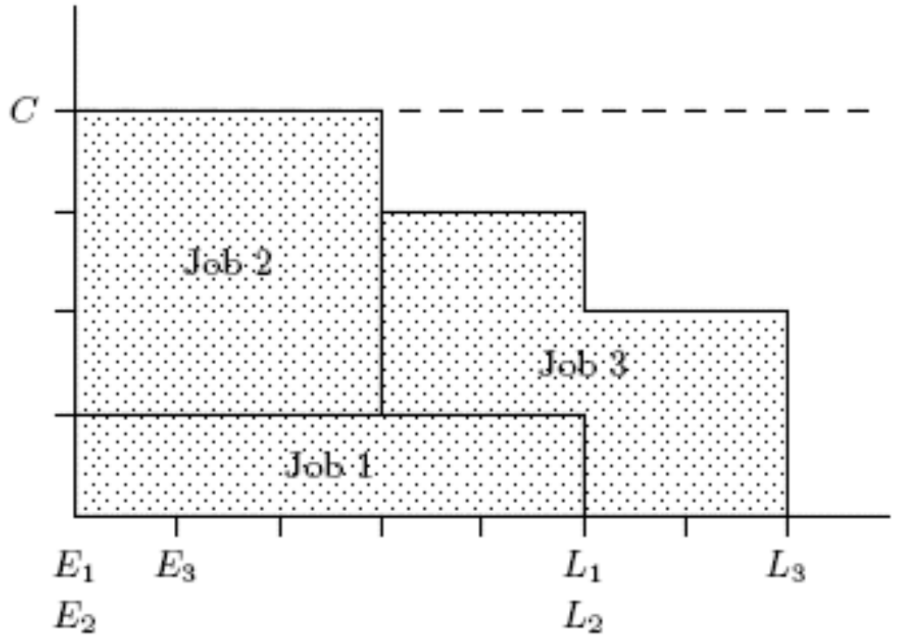


Edge finding for cumulative scheduling

We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

Because the total **energy** required exceeds the area between the earliest release time and the later deadline of jobs 1,2:

$$e_3 + e_{\{1,2\}} > C \cdot \left(L_{\{1,2\}} - E_{\{1,2,3\}} \right)$$



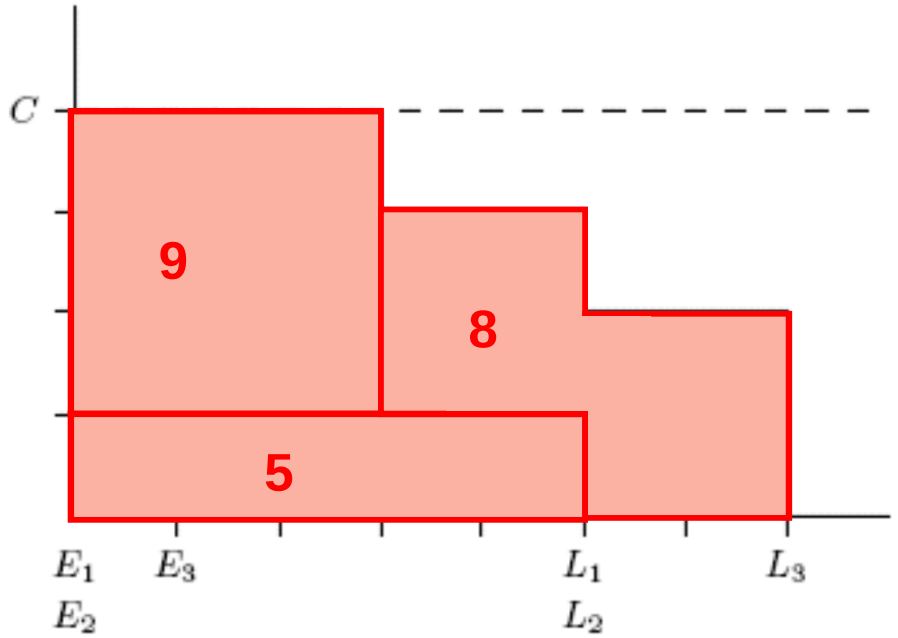
Edge finding for cumulative scheduling

We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

Because the total **energy** required exceeds the area between the earliest release time and the later deadline of jobs 1,2:

$$e_3 + e_{\{1,2\}} > C \cdot \left(L_{\{1,2\}} - E_{\{1,2,3\}} \right)$$

Total energy required = 22



Edge finding for cumulative scheduling

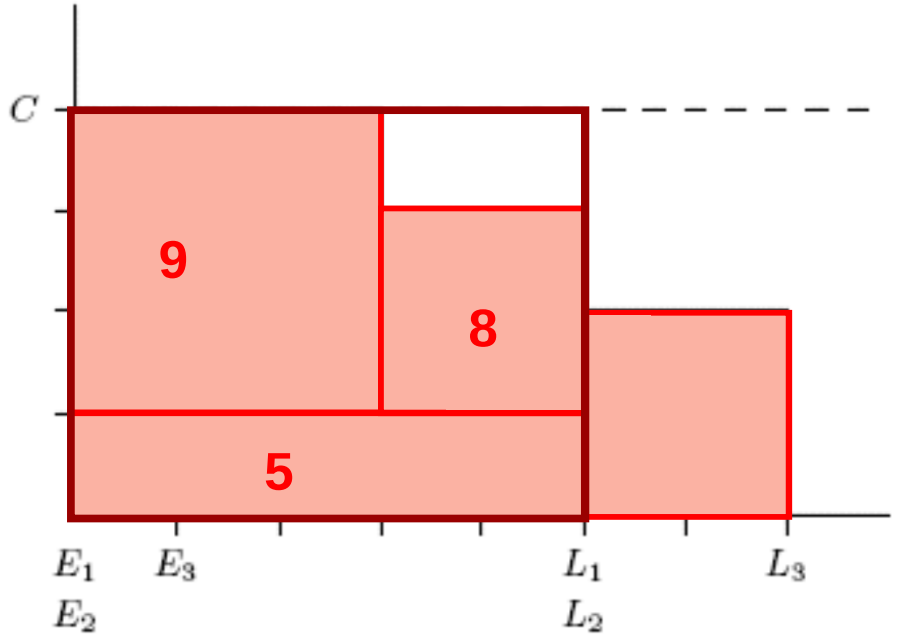
We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

Because the total **energy** required exceeds the area between the earliest release time and the later deadline of jobs 1,2:

$$e_3 + e_{\{1,2\}} > C \cdot \left(L_{\{1,2\}} - E_{\{1,2,3\}} \right)$$

Total energy required = 22

Area available = 20



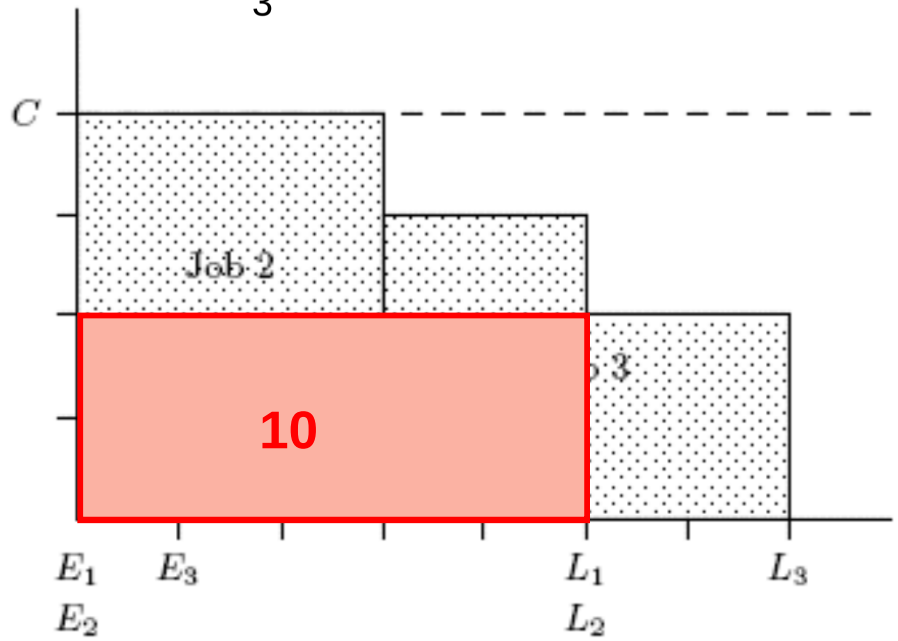
Edge finding for cumulative scheduling

We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

We can update the release time of job 3 to

$$E_{\{1,2\}} + \frac{e_J - (C - c_3)(L_{\{1,2\}} - E_{\{1,2\}})}{c_3}$$

Energy available for jobs 1,2 if space is left for job 3 to start anytime = 10



Edge finding for cumulative scheduling

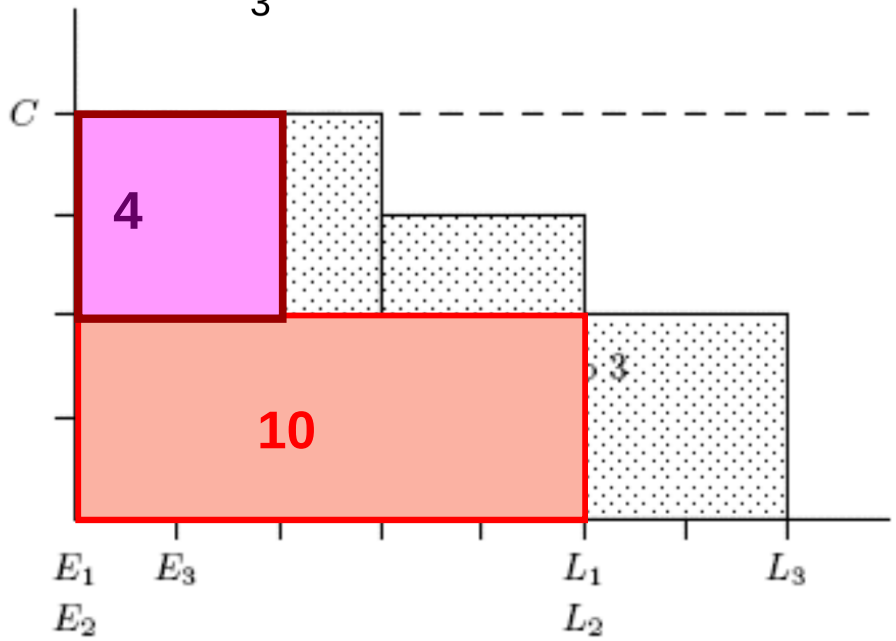
We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

We can update the release time of job 3 to

$$E_{\{1,2\}} + \frac{e_J - (C - c_3)(L_{\{1,2\}} - E_{\{1,2\}})}{c_3}$$

Energy available for jobs 1,2 if space is left for job 3 to start anytime = 10

Excess energy required by jobs 1,2 = 4



Edge finding for cumulative scheduling

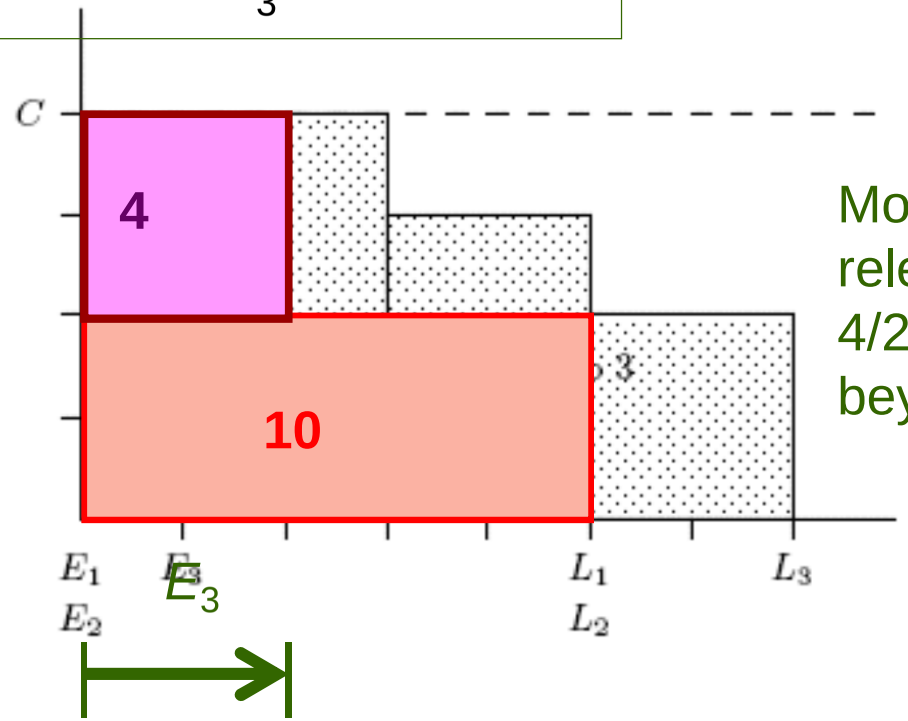
We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

We can update the release time of job 3 to

$$E_{\{1,2\}} + \frac{e_J - (C - c_3)(L_{\{1,2\}} - E_{\{1,2\}})}{c_3}$$

Energy available for jobs 1,2 if space is left for job 3 to start anytime = 10

Excess energy required by jobs 1,2 = 4



Move up job 3 release time 4/2 = 2 units beyond $E_{\{1,2\}}$

Edge finding for cumulative scheduling

In general, if $e_{J\cup\{k\}} > C \cdot \left(L_J - E_{J\cup\{k\}}\right)$

then $k > J$, and update E_k to

$$\max_{\substack{J'\subset J \\ e_{J'}-(C-c_k)(L_{J'}-E_{J'})>0}} \left\{ E_{J'} + \frac{e_{J'} - (C-c_k)(L_{J'}-E_{J'})}{c_k} \right\}$$

In general, if $e_{J\cup\{k\}} > C \cdot \left(L_{J\cup\{k\}} - E_J\right)$

then $k < J$, and update L_k to

$$\min_{\substack{J'\subset J \\ e_{J'}-(C-c_k)(L_{J'}-E_{J'})>0}} \left\{ L_{J'} - \frac{e_{J'} - (C-c_k)(L_{J'}-E_{J'})}{c_k} \right\}$$

Edge finding for cumulative scheduling

Key result:

There is an $O(n^2)$ algorithm that finds all applications of the edge finding rules.

Nuitjen & Aarts 1994, 1996
Baptiste, Le Pape & Nuitjen 2001

Other propagation rules for cumulative scheduling

- Extended edge finding.
- Timetabling.
- Not-first/not-last rules.
- Energetic reasoning.

Planning and cumulative scheduling

- Assign jobs to facilities (e.g. factories), with cumulative scheduling in each facility.

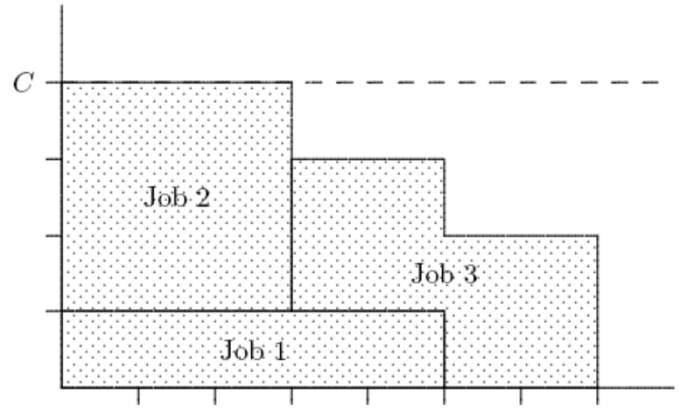
Planning and cumulative scheduling

- Assign jobs to facilities (e.g. factories), with cumulative scheduling in each facility.
- Benders cut for minimum **makespan**, with a **cumulative scheduling** subproblem:

$$M \geq M_i^* - \left(\sum_{j \in J_i} p_{ij}(1 - x_{ij}) + \max_{j \in J_i}\{d_j\} - \min_{j \in J_i}\{d_j\} \right)$$

Jobs currently assigned to machine *i*

Minimum makespan on machine *i* for jobs currently assigned



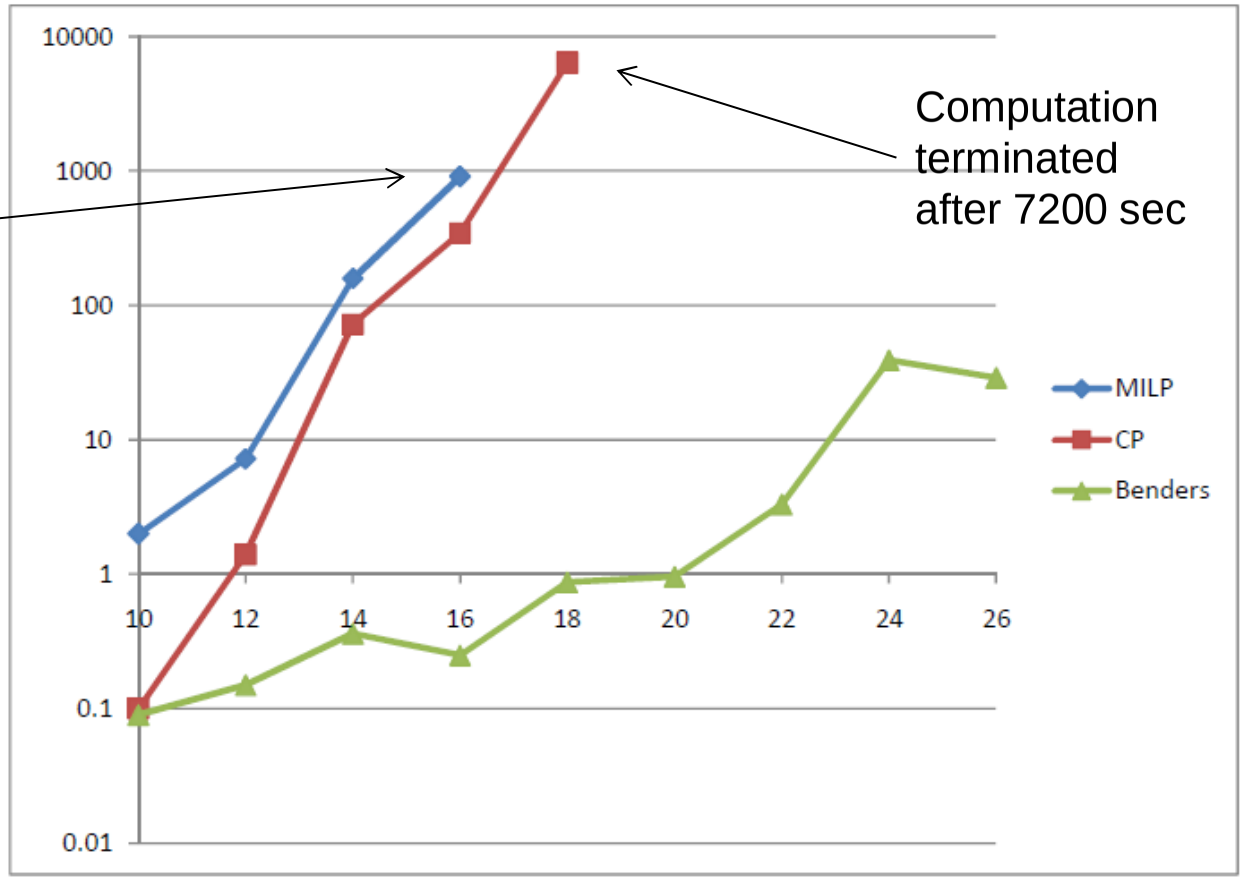
Hooker 2004

Planning and cumulative scheduling

Minimum Cost

Computation time (sec) vs. number of jobs

4 facilities (each point = 5 instances)

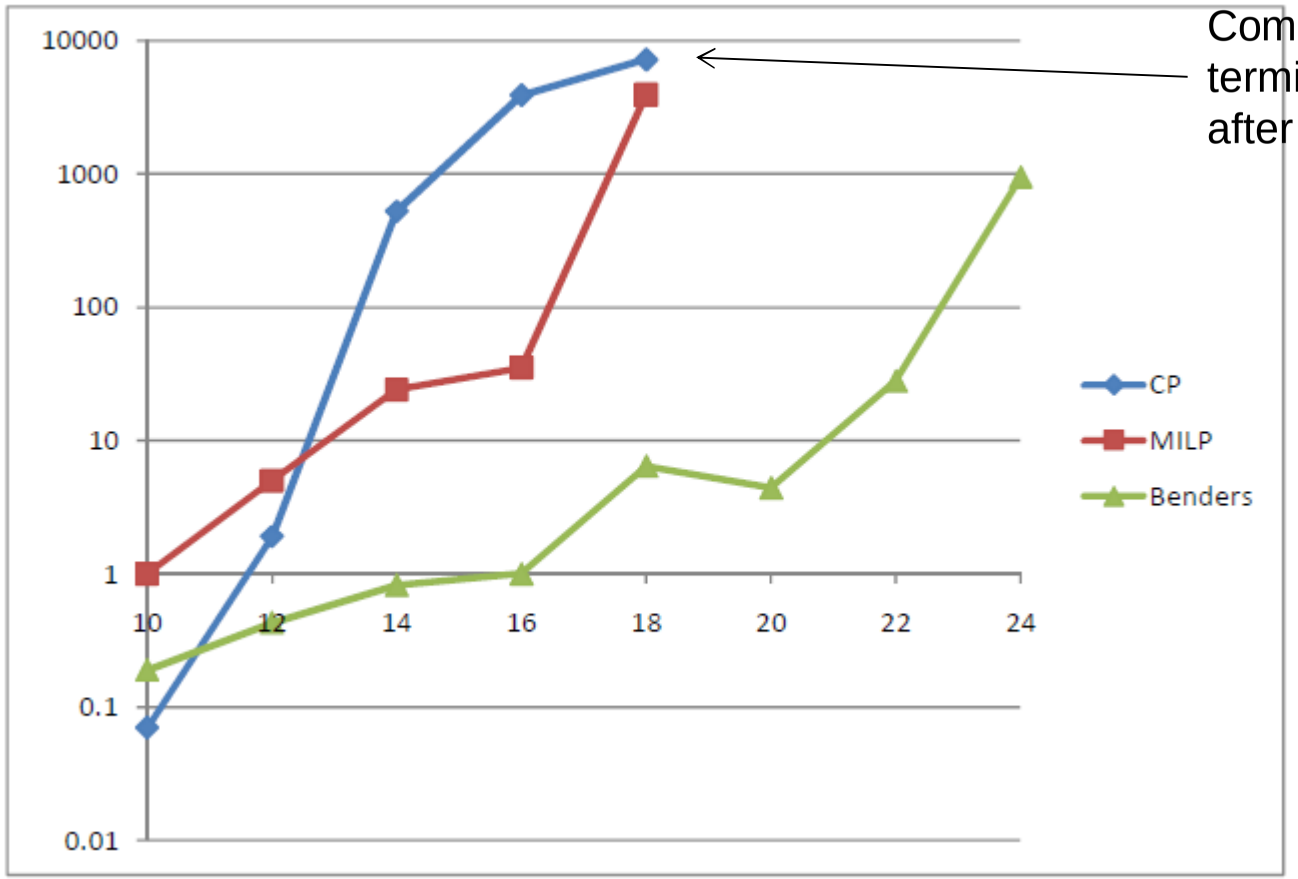


Planning and cumulative scheduling

Minimum Makespan

Computation time (sec) vs. number of jobs

4 facilities (each point = 5 instances)



Other applications of logic-based Benders

- Steel production scheduling
- Batch scheduling in a chemical plant (BASF)
- Assembly line scheduling (Citroën/Pugeot)
- Location-allocation
- Dispatching of automated guided vehicles
- Queuing design & control
- Real-time scheduling of computer processes
- Traffic diversion
- Home health care delivery*
- Sports scheduling

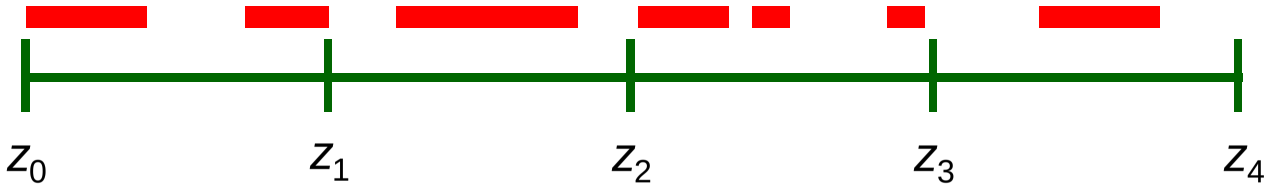
*Cire & Hooker 2010

Single-facility Scheduling

- Can we apply decomposition to pure scheduling problems?
- Consider single-machine scheduling with a **long time horizon**.

Single-facility Scheduling

- Can we apply decomposition to pure scheduling problems?
- Consider single-machine scheduling with a **long time horizon**.
- **Segmented problem**: jobs must finish within a segment.
  - Segment boundaries are weekends or shutdown times.
  - Master problem assigns jobs to segments.



Coban & Hooker 2010

Single-facility scheduling

- Benders cut from segment *i* (min makespan problem):
  -

$$M \geq M_i^* - \sum_{j \in J_i'} p_j \left(1 - y_{ij}\right) - w_i - M_i^* \sum_{j \in J_i''} \left(1 - y_{ij}\right)$$

$$q_i \leq 1 - y_{ij}, \quad j \in J_i$$

$$w_i \leq \left(\max_{j \in J_i'} \{d_j\} - \min_{j \in J_i'} \{d_j\} \right) \sum_{j \in J_i'} \left(1 - y_{ij}\right)$$

$$w_i \leq \max_{j \in J_i'} \{d_j\} - \min_{j \in J_i'} \{d_j\}$$

y_{ij} = 1 when job *j* is assigned to segment *i*

J_i' = { jobs with release times before start of segment *i* }

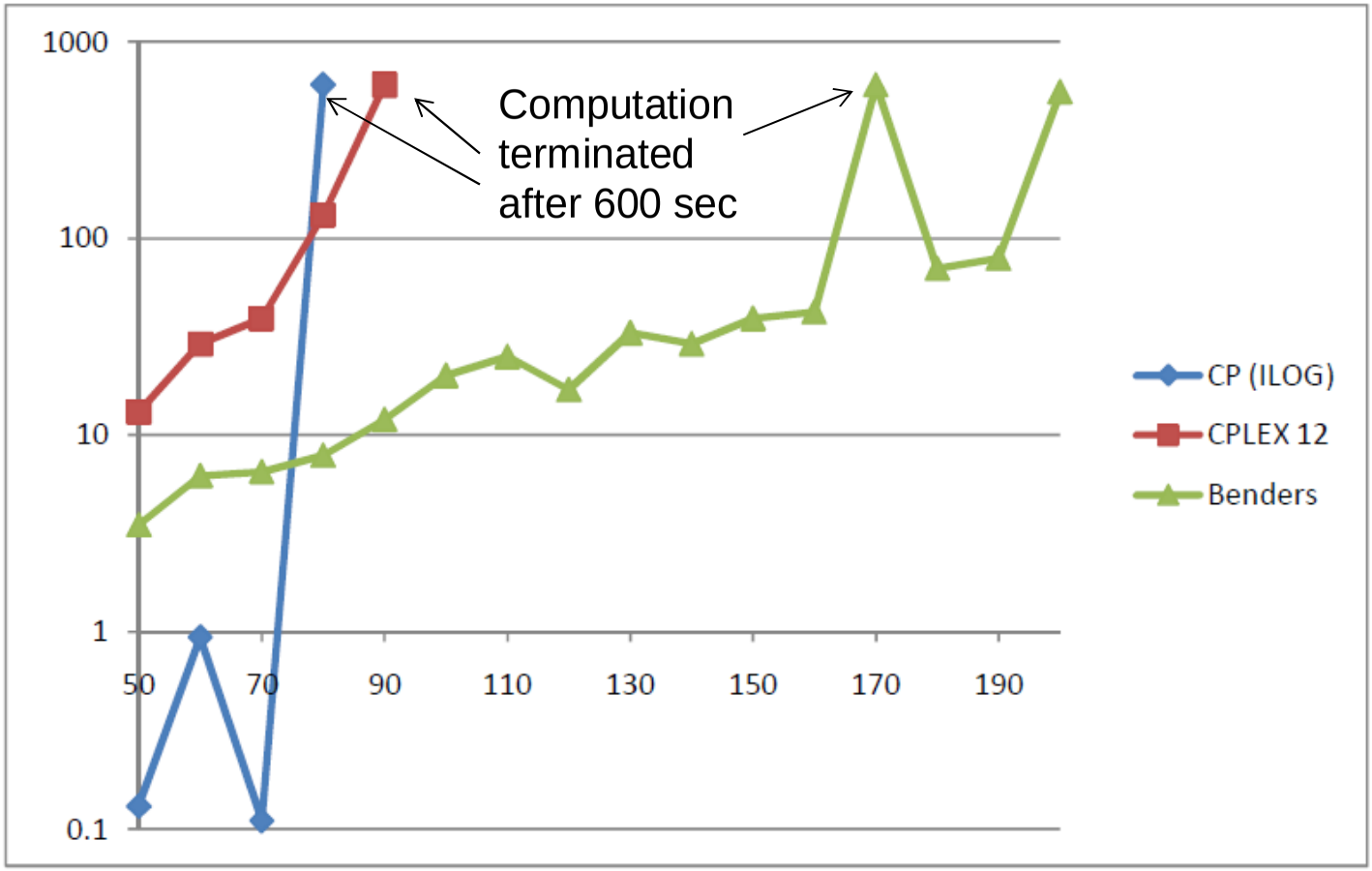
J_i'' = { jobs with release times after start of segment *i* }

Single-facility scheduling

Segmented Problem

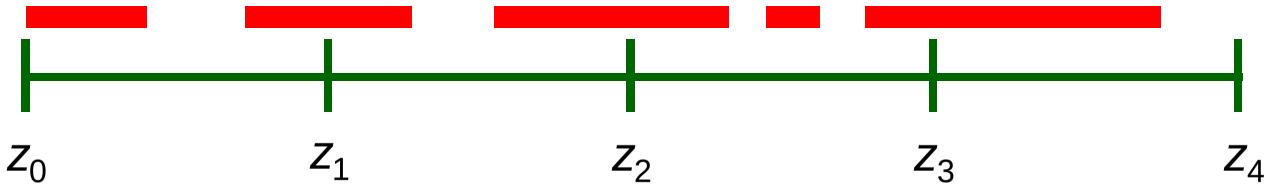
Computation time (sec) vs. number of jobs

Time horizon proportional to no. of jobs



Single-facility scheduling

- **Unsegmented problem**: jobs can be scheduled anytime within their time windows.
  - Segment boundaries are only for decomposition purposes.



Coban & Hooker 2011

Single-facility scheduling

- Master problem must encode whether a job is split between segments.
  -

$$\min M$$

$$\sum_i y_{ij} \geq 1, \quad \text{all } j$$

$$y_{ij} = y_{ij0} + y_{ij1} + y_{ij2} + y_{ij3}, \quad \text{all } i, j$$

$$\sum_j y_{ij1} \leq 1, \quad \sum_j y_{ij2} \leq 1, \quad \sum_j y_{ij3} \leq 1,$$

$$y_{ij1} \leq y_{i-1,j,2} + y_{i-1,j,3}, \quad \text{all } i > 1, \text{ all } j$$

$$y_{ij2} \leq y_{i-1,j,1} + y_{i-1,j,3}, \quad \text{all } i < n, \text{ all } j$$

$$y_{ij3} \leq y_{i-1,j,3} + y_{i-1,j,2}, \quad \text{all } i > 1, \text{ all } j$$

$$y_{ij3} \leq y_{i+1,j,3} + y_{i+1,j,1}, \quad \text{all } i < n, \text{ all } j$$

$$\sum_i y_{ij0} \leq 1, \quad \sum_i y_{ij1} \leq 1, \quad \sum_i y_{ij2} \leq 1,$$

$$y_{ij1} = y_{ij3} = y_{nj2} = y_{nj3} = 0, \quad \text{all } j$$

$$\sum_i y_{ij3} \leq \left\lfloor \frac{p_j}{z_{i+1} - z_i} \right\rfloor, \quad \text{all } j$$

Benders cuts, relaxation

$$y_{ij}, y_{ijk} \in \{0,1\}$$

Single-facility scheduling

- Benders cuts are generated for several cases.
  - Benders cut when a job j_1 completes at start of segment and some job starts at its release time:

$$M \geq M_i^* (1-\eta_i) - M_i^* \sum_{j \in J_i} (1-y_{ij})$$

$$\bar{x}_{ij_1} - x_{ij_1} + \Delta_i \leq p_{\min} + (\bar{x}_{ij_1} + \Delta_i - p_{\min})\eta_i - \varepsilon$$

$$\bar{x}_{ij_1} - x_{ij_1} + \Delta_i \geq p_{\min} - (p_{j_1} - x_{ij_1} + p_{\min} - \Delta_i)(1-\eta_i)$$

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

= 0 when reoptimizing schedule on segment doesn't change job order

Start time of first job k for which $s_k^* = r_k$

Start time of job j in solution of subproblem

Release time

$$\delta^* = \min_{j \in J_{i0}} \{ s_j^* - r_{ij} \mid s_j^* < s_k^* \}$$

$$p_{\min} = \min_{j \in J_{i0}} \{ p_j \mid s_j^* > s_k^*, r_j + p_j \leq r_k, p_j \leq \Delta_i + \delta^* \}$$

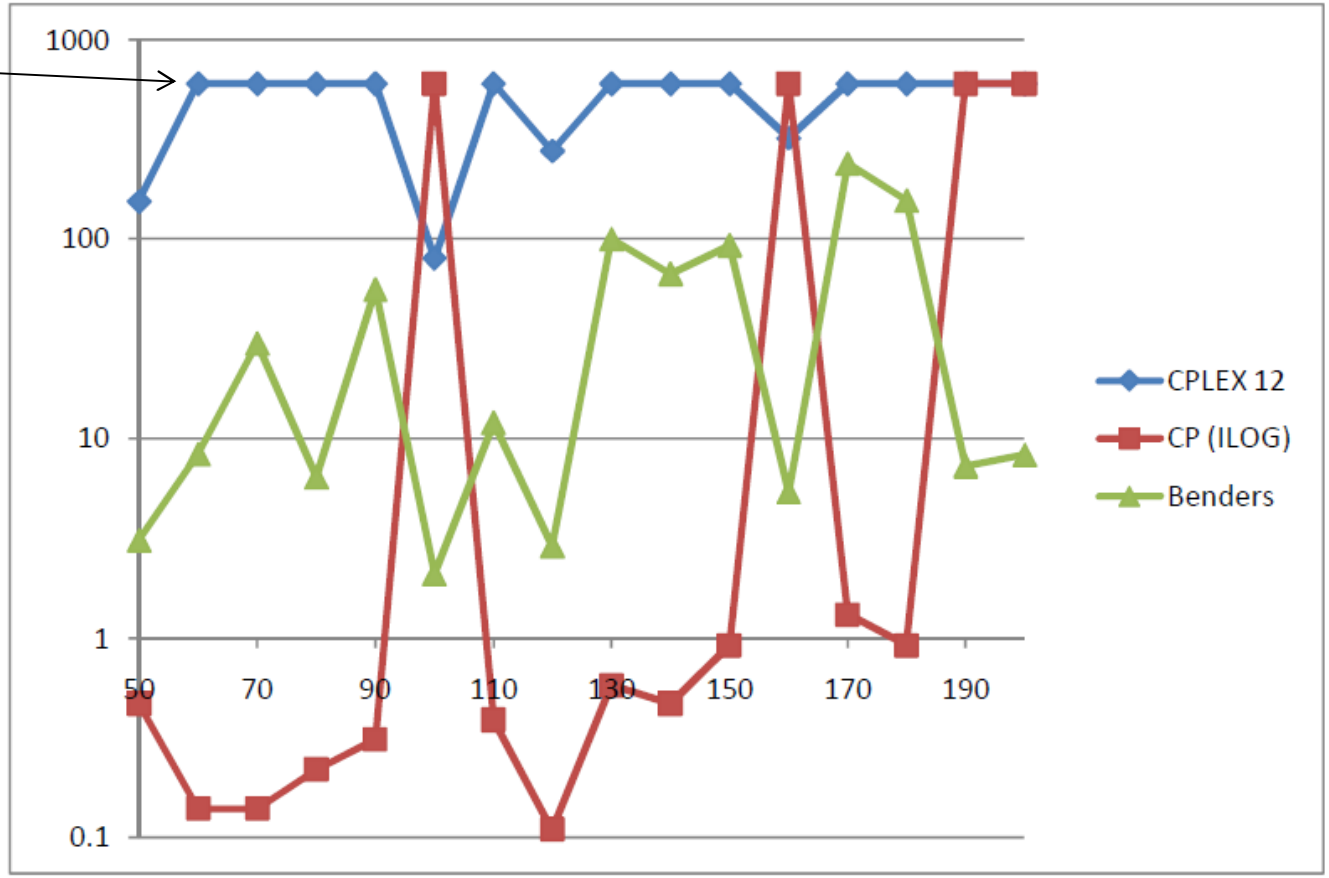
Single-facility scheduling

Unsegmented Problem

Computation time (sec) vs. number of jobs

Time horizon proportional to no. of jobs

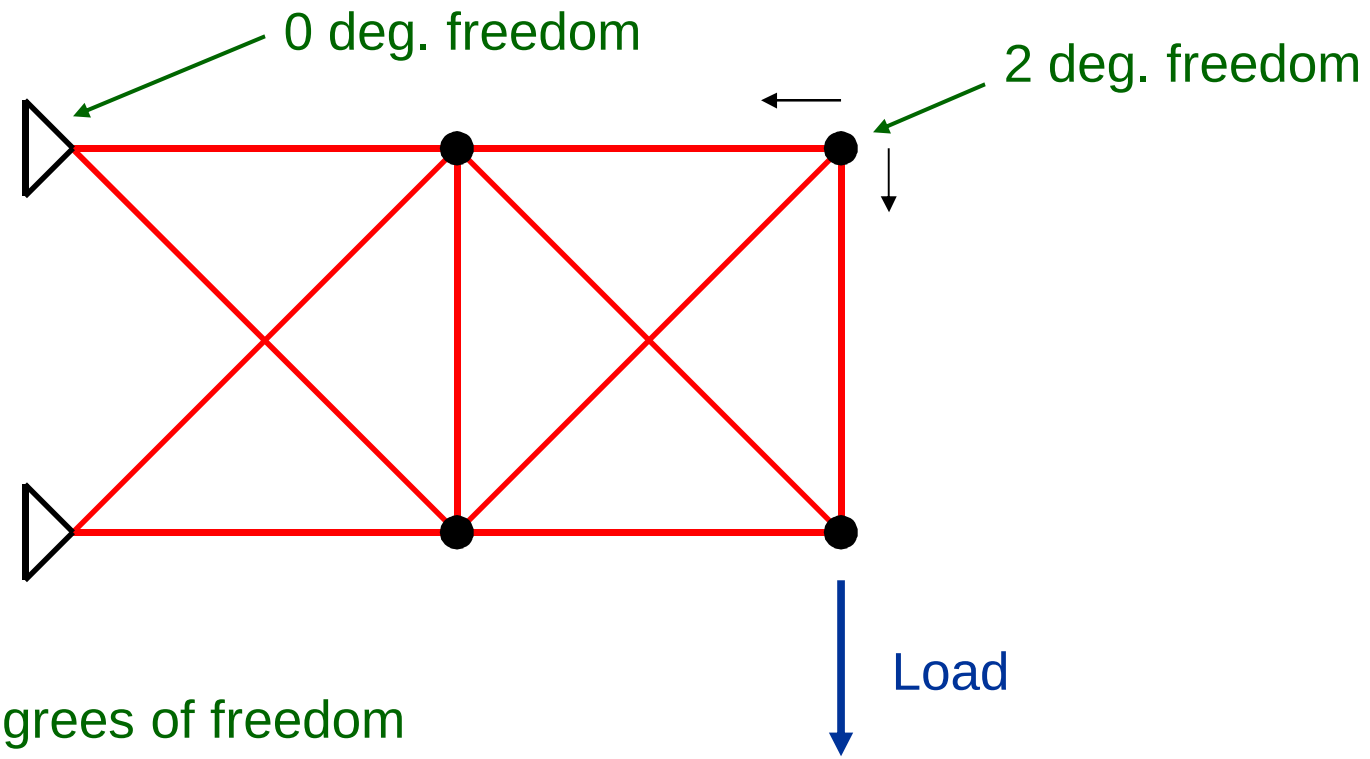
Computation terminated after 600 sec



Truss Structure Design

Select size of each bar (possibly zero) to support the load while minimizing weight.

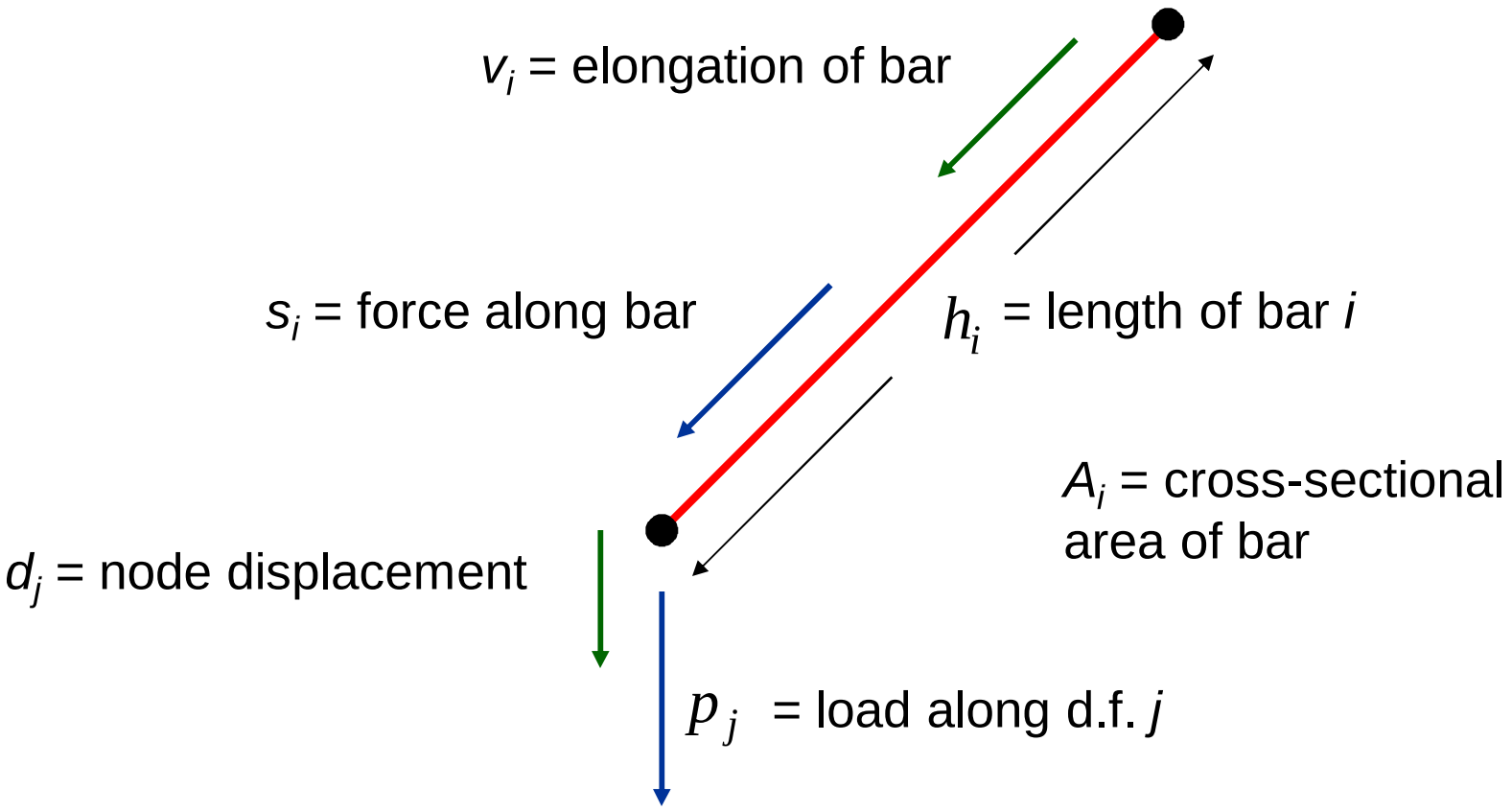
10-bar cantilever truss



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

Truss Structure Design

Integrated approach

• Search: Branch by **splitting** the range of areas A_i (no need for 0-1 variables).

• Relaxation: Generate a **quasi-relaxation**, which is linear and much smaller than MILP model.

• Inference: Use **logic cuts**.

Bollapragada, Ghattas & Hooker 2001

Truss Structure Design

Theorem

Suppose we minimize cx subject to $g(x,y) \leq 0$.

If $g(x,y)$ is semihomogeneous in $x \in R^n$ and concave in scalar y, then the following is a **quasi-relaxation** of $g(x,y) \leq 0$:

$$g(x^1, y_L) + g(x^2, y_U) \leq 0$$
$$\alpha x^L \leq x^1 \leq \alpha x^U$$
$$(1-\alpha)x^L \leq x^2 \leq (1-\alpha)x^U$$
$$x = x^1 + x^2$$

Hooker 2005

Truss Structure Design

Theorem

Suppose we minimize cx subject to $g(x,y) \leq 0$.

If $g(x,y)$ is semihomogeneous in $x \in R^n$ and concave in scalar y, then the following is a **quasi-relaxation** of $g(x,y) \leq 0$:

$$g(x^1, y_L) + g(x^2, y_U) \leq 0$$
$$\alpha x^L \leq x^1 \leq \alpha x^U$$
$$(1-\alpha)x^L \leq x^2 \leq (1-\alpha)x^U$$
$$x = x^1 + x^2$$

Hooker 2005

Not a valid relaxation, and it may contain different variables, But its optimal value is a **valid lower bound** on the optimal value of the original problem.

Truss Structure Design

Theorem

Suppose we minimize cx subject to $g(x,y) \leq 0$.

If $g(x,y)$ is semihomogeneous in $x \in R^n$ and concave in scalar y, then the following is a **quasi-relaxation** of $g(x,y) \leq 0$:

$$g(x^1, y_L) + g(x^2, y_U) \leq 0$$
$$\alpha x^L \leq x^1 \leq \alpha x^U$$
$$(1-\alpha)x^L \leq x^2 \leq (1-\alpha)x^U$$
$$x = x^1 + x^2$$

Hooker 2005

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

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

Truss Structure Design

Theorem

Suppose we minimize cx subject to $g(x,y) \leq 0$.

If $g(x,y)$ is semihomogeneous in $x \in R^n$ and concave in scalar y, then the following is a **quasi-relaxation** of $g(x,y) \leq 0$:

$$g(x^1, y_L) + g(x^2, y_U) \leq 0$$
$$\alpha x^L \leq x^1 \leq \alpha x^U$$
$$(1-\alpha)x^L \leq x^2 \leq (1-\alpha)x^U$$
$$x = x^1 + x^2$$

Bounds on y

Hooker 2005

Truss Structure Design

Theorem (JNH 2005)

Suppose we minimize cx subject to $g(x,y) \leq 0$.

If $g(x,y)$ is semihomogeneous in $x \in R^n$ and concave in scalar y, then the following is a **quasi-relaxation** of $g(x,y) \leq 0$:

$$g(x^1, y_L) + g(x^2, y_U) \leq 0$$
$$\alpha x^L \leq x^1 \leq \alpha x^U$$
$$(1-\alpha)x^L \leq x^2 \leq (1-\alpha)x^U$$
$$x = x^1 + x^2$$

Bounds on x

Hooker 2005

Truss Structure Design

Theorem (JNH 2005)

Suppose we minimize cx subject to $g(x,y) \le 0$.

If $g(x,y)$ is semihomogeneous in $x \in R^n$ and concave in scalar y, then the following is a **quasi-relaxation** of $g(x,y) \le 0$:

$$g(x^1, y_L) + g(x^2, y_U) \le 0$$
$$\alpha x^L \le x^1 \le \alpha x^U$$
$$(1-\alpha)x^L \le x^2 \le (1-\alpha)x^U$$
$$x = x^1 + x^2$$

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

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

Truss Structure Design

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

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

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

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

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

Hooke's law is linearized

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

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

Elongation bounds split into 2 sets of bounds

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

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

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

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

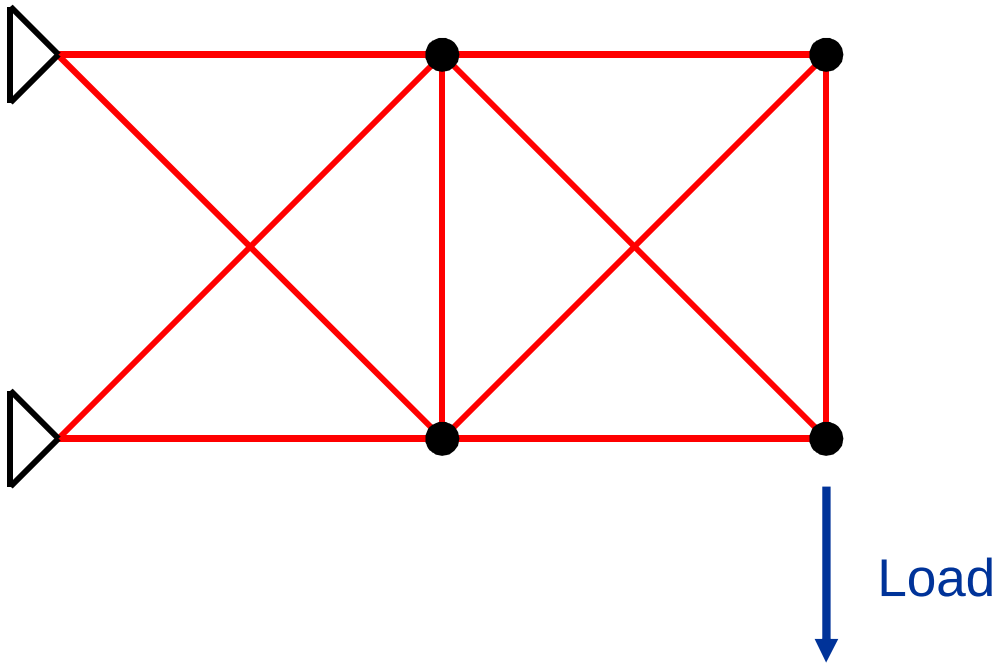
Truss Structure Design

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 }
```

Truss Structure Design

10-bar cantilever truss



Truss Structure Design

Yunes, Aron & Hooker 2010

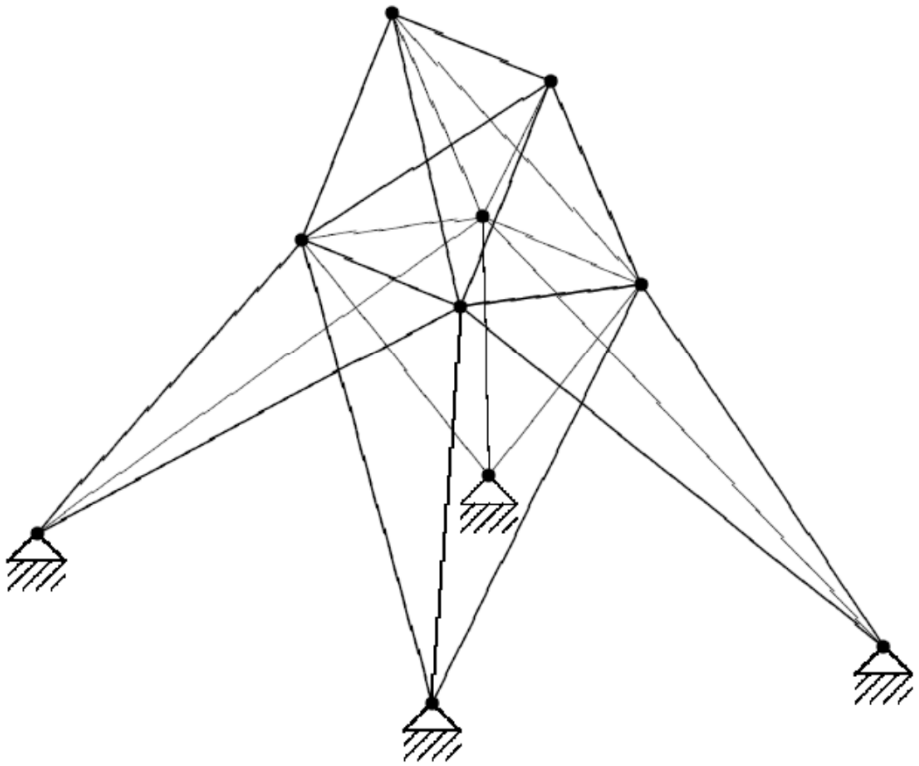
Computational results (seconds)

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

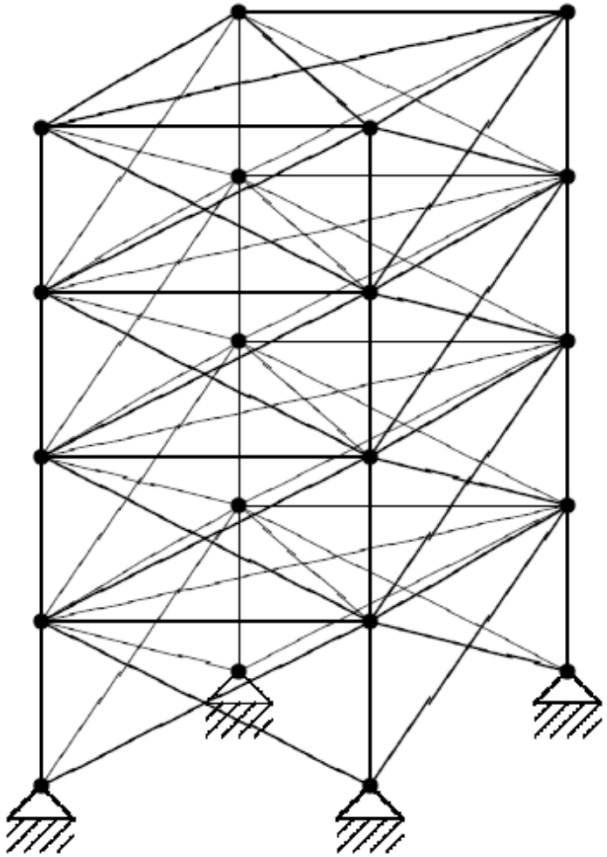
Truss Structure Design

25-bar problem



Truss Structure Design

72-bar problem



Truss Structure Design

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

Summary

- Combinatorial optimizations methods tend to use a common **primal/dual/dual** approach.
  - This can be a basis for a **general-purpose solver**
  - …if the problem is modeled with **metaconstraints**.

Summary

- Combinatorial optimizations methods tend to use a common **primal/dual/dual** approach.
  - This can be a basis for a **general-purpose solver**.
  - …if the problem is modeled with **metaconstraints**.

- This more general point of view can suggest new methods.
  - Methods based on **inference duality** illustrated here.
  - So far, new methods based on **relaxation duality** are primarily discrete
  - …and/or trees, mini-buckets & nonserial DP, multivalued decision diagrams, etc.

Gracias.

¿Hay preguntas?