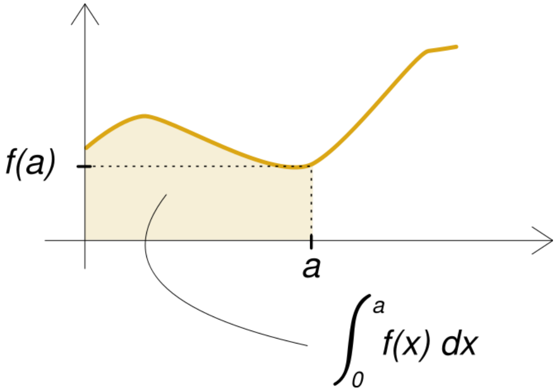
Tutorial
Hybrid Mixed-Integer Programming and Constraint Programing Methods

John Hooker
Carnegie Mellon University

CIMI, Toulouse
June 2018



Why Integrate CP and MIP?

Complementary Strengths

Outline of the Tutorial

Complementary Strengths

- CP:
  - Inference methods
  - Modeling
  - Exploits local structure
- MIP:
  - Relaxation methods
  - Duality theory
  - Exploits global structure

Let’s bring them together!

Comparison

CP vs. MIP

CP	MIP
Logic processing	Numerical calculation
Inference (filtering, constraint propagation)	Relaxation
High-level modeling (global constraints)	Atomistic modeling (linear inequalities)
Branching	Branching
Constraint-based processing	Independence of model And algorithm

Programming $\neq$ programming

- In **constraint programming**:
  - *programming* = a form of computer programming (constraint-based processing)
- In **mathematical programming**:
  - *programming* = logistics planning (historically)

CP vs. MIP

- In **CP**, each constraint invokes a procedure that screens out unacceptable solutions.
  - Much as each line of a computer program invokes an operation.

- In **MIP**, equations (constraints) describe the problem but don't tell how to solve it.

Advantages of CP

- Better at sequencing and scheduling
  - …where MP methods have weak relaxations.
- Adding messy constraints makes the problem easier.
  - The more constraints, the better.
- More powerful modeling language.
  - Global constraints lead to succinct models.
  - Constraints convey problem structure to the solver.
- "Better at highly-constrained problems"
  - Misleading – better when constraints propagate well, or when constraints have few variables.

Advantages of MIP

- Deals naturally with continuous variables.
    - Continuous relaxation, numerical techniques
- Handles constraints with many variables.
    - These constraints don’t propagate well in CP.
- Good at finding optimal (as opposed to feasible) solutions.
    - Sophisticated relaxation technology provides bounds.
- Scales up
    - Decades of engineering, orders of magnitude speedup

Obvious solution…

- Integrate CP and MIP.

Obvious solution…

- Integrate CP and MIP.

Two basic strategies…

- Combine CP and MIP in a single solution method.
- Link CP and MIP solvers in a principled way.

Outline of the Tutorial

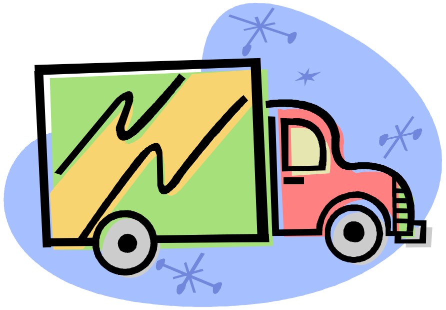
- Why Integrate OR and CP?
- Combine CP and MIP in a single solution method
  - Designing an Integrated Solver
  - Linear Relaxation and Duality
  - Mixed Integer/Linear Modeling
  - Cutting Planes
  - Lagrangean Relaxation and CP
- Link CP and MIP solvers
  - Constraint Programming Concepts
  - CP Filtering Algorithms
  - CP-based Branch and Price
  - Benders Decomposition
- Software

Hybrid methods I am leaving out

- CP and dynamic programming
- OR-based filtering methods (e.g. flow models, edge finding)
- Decision diagrams (to be presented by W-J van Hoeve)
- CP and local search (to be presented by Paul Shaw)

Background Reading

- J. N. Hooker and W.-J. van Hoeve, Constraint programming and operations research, *Constraints* **23** (2018) 172-195. Contains many references.
- J. N. Hooker, *Integrated Methods for Optimization*, 2nd ed., Springer (2012). Contains many exercises.



Initial Example: Designing an Integrated Solver

Freight Transfer

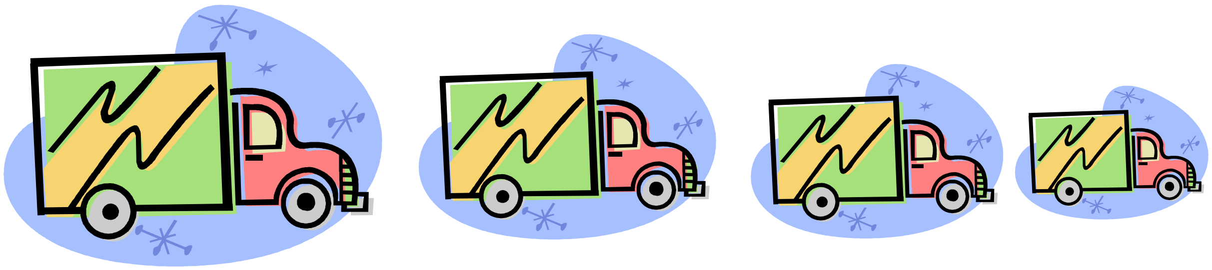
Bounds Propagation

Cutting Planes

Branch-infer-and-relax Tree

Example: Freight Transfer

- Transport 42 tons of freight using 8 trucks, which come in 4 sizes…



Truck size	Number available	Capacity (tons)	Cost per truck
1	3	7	90
2	3	5	60
3	3	4	50
4	3	3	40

Number of trucks of type 1

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

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

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

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

Knapsack covering constraint

Knapsack packing constraint

Truck type	Number available	Capacity (tons)	Cost per truck
1	3	7	90
2	3	5	60
3	3	4	50
4	3	3	40

Bounds propagation

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

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

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

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

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

Bounds propagation

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

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

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

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

Reduced domain

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

Bounds consistency

- Let $\{L_j, \ldots, U_j\}$ be the domain of x_j
- A constraint set is **bounds consistent** if for each j :
  - $x_j = L_j$ in some feasible solution and
  - $x_j = U_j$ in some feasible solution.
- Bounds consistency $\Rightarrow$ we will not set x_j to any infeasible values during branching.
- Bounds propagation achieves bounds consistency for a **single inequality**.
  - $7x_1 + 5x_2 + 4x_3 + 3x_4 \geq 42$ is bounds consistent when the domains are $x_1 \in \{1,2,3\}$ and $x_2, x_3, x_4 \in \{0,1,2,3\}$.
- But not necessarily for a **set** of inequalities.

Bounds consistency

- Bounds propagation may not achieve bounds consistency for a set of constraints.
- Consider set of inequalities

$$x_1 + x_2 \geq 1$$
$$x_1 - x_2 \geq 0$$

with domains x_1, $x_2 \in \{0,1\}$, solutions $(x_1,x_2) = (1,0), (1,1)$.

- Bounds propagation has no effect on the domains.
- But constraint set is not bounds consistent because $x_1 = 0$ in no feasible solution.

Cutting Planes

Begin with continuous relaxation

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

Replace domains with bounds

This is a linear programming problem, which is easy to solve.

Its optimal value provides a lower bound on optimal value of original problem.

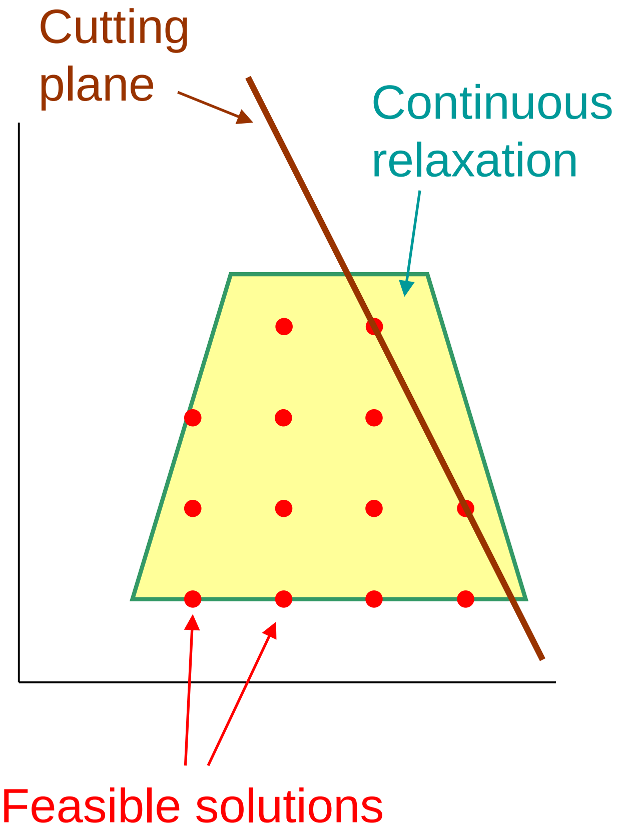
Cutting planes (valid inequalities)

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

We can create a **tighter** relaxation (larger minimum value) with the addition of **cutting planes**.

Cutting planes (valid inequalities)

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



All feasible solutions of the original problem satisfy a cutting plane (i.e., it is **valid**).

But a cutting plane may exclude ("**cut off**") solutions of the continuous relaxation.

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 $x_1 = x_2 = 3$.

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

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)

Let x_i have domain $[L_i, U_i]$ and let $a \geq 0$.

In general, a **packing** P for $ax \geq a_0$ satisfies

$$\sum_{i \notin P} a_i x_i \geq a_0 - \sum_{i \in P} a_i U_i$$

and generates a **knapsack cut**

$$\sum_{i \notin P} x_i \geq \left\lceil \frac{a_0 - \sum_{i \in P} a_i U_i}{\max_{i \notin P} \{a_i\}} \right\rceil$$

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.

Branch-infer-and-relax tree

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

Propagate bounds and solve relaxation of original problem.

Branch-infer-and-relax tree

Branch on a variable with nonintegral value in the relaxation.

$x_1 \in \{\ \ 123\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (2⅓,3,2⅔,0)$
value = 523⅓

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

Branch-infer-and-relax tree

Propagate bounds and solve relaxation.

Since relaxation is infeasible, backtrack.

$x_1 \in \{\ \ 123\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (2⅓,3,2⅔,0)$
value = 523⅓

$x_1 \in \{1,2\}$ | $x_1 = 3$

$x_1 \in \{\ \ 12\ \ \}$
$x_2 \in \{\ \ \ \ 23\}$
$x_3 \in \{\ \ 123\}$
$x_4 \in \{\ \ 123\}$
infeasible relaxation

Branch-infer-and-relax tree

Propagate bounds and solve relaxation.

Branch on nonintegral variable.

$x_1 \in \{\ \ 123\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (2⅓,3,2⅔,0)$
value = 523⅓

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

$x_1 \in \{\ \ 12\ \ \}$
$x_2 \in \{\ \ \ \ 23\}$
$x_3 \in \{\ \ 123\}$
$x_4 \in \{\ \ 123\}$
infeasible relaxation

$x_1 = 3$

$x_1 \in \{\ \ \ \ \ \ 3\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (3,2.6,2,0)$
value = 526

$x_2 \in \{0,1,2\}$

$x_2 = 3$

Branch-infer-and-relax tree

Branch again.

$x_1 \in \{\ \ 123\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (2⅓,3,2⅔,0)$
value = 523⅓

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

$x_1 \in \{\ \ 12\ \ \}$
$x_2 \in \{\ \ \ \ 23\}$
$x_3 \in \{\ \ 123\}$
$x_4 \in \{\ \ 123\}$
infeasible relaxation

$x_1 = 3$

$x_1 \in \{\ \ \ \ \ \ 3\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (3,2.6,2,0)$
value = 526

$x_2 \in \{0,1,2\}$

$x_1 \in \{\ \ \ \ \ \ 3\}$
$x_2 \in \{012\ \ \}$
$x_3 \in \{\ \ 123\}$
$x_4 \in \{0123\}$
$x = (3,2,2¾,0)$
value = 527½

$x_3 \in \{1,2\}$

$x_3 = 3$

$x_2 = 3$

Branch-infer-and-relax tree

$x_1 \in \{\ 123\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (2⅓,3,2⅔,0)$
value = 523⅓

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

$x_1 \in \{\ 12\ \}$
$x_2 \in \{\ \ 23\}$
$x_3 \in \{\ 123\}$
$x_4 \in \{\ 123\}$
infeasible relaxation

$x_1 = 3$

$x_1 \in \{\ \ \ 3\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (3,2.6,2,0)$
value = 526

$x_2 \in \{0,1,2\}$

$x_1 \in \{\ \ \ 3\}$
$x_2 \in \{012\ \}$
$x_3 \in \{\ 123\}$
$x_4 \in \{0123\}$
$x = (3,2,2¾,0)$
value = 527½

$x_2 = 3$

$x_3 \in \{1,2\}$

$x_1 \in \{\ \ \ 3\}$
$x_2 \in \{\ 12\ \}$
$x_3 \in \{\ 12\ \}$
$x_4 \in \{\ 123\}$
$x = (3,2,2,1)$
value = 530
feasible solution

$x_3 = 3$

Solution of relaxation is integral and therefore feasible in the original problem.

This becomes the **incumbent solution**.

Branch-infer-and-relax tree

Solution is nonintegral, but we can backtrack because value of relaxation is no better than incumbent solution.

$x_1 \in \{\ 123\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (2⅓,3,2⅔,0)$
value = 523⅓

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

$x_1 \in \{\ 12\ \}$
$x_2 \in \{\ \ 23\}$
$x_3 \in \{\ 123\}$
$x_4 \in \{\ 123\}$
infeasible relaxation

$x_1 = 3$

$x_1 \in \{\ \ \ 3\}$
$x_2 \in \{0123\}$
$x_3 \in \{0123\}$
$x_4 \in \{0123\}$
$x = (3,2.6,2,0)$
value = 526

$x_2 \in \{0,1,2\}$

$x_1 \in \{\ \ \ 3\}$
$x_2 \in \{012\ \}$
$x_3 \in \{\ 123\}$
$x_4 \in \{0123\}$
$x = (3,2,2¾,0)$
value = 527½

$x_2 = 3$

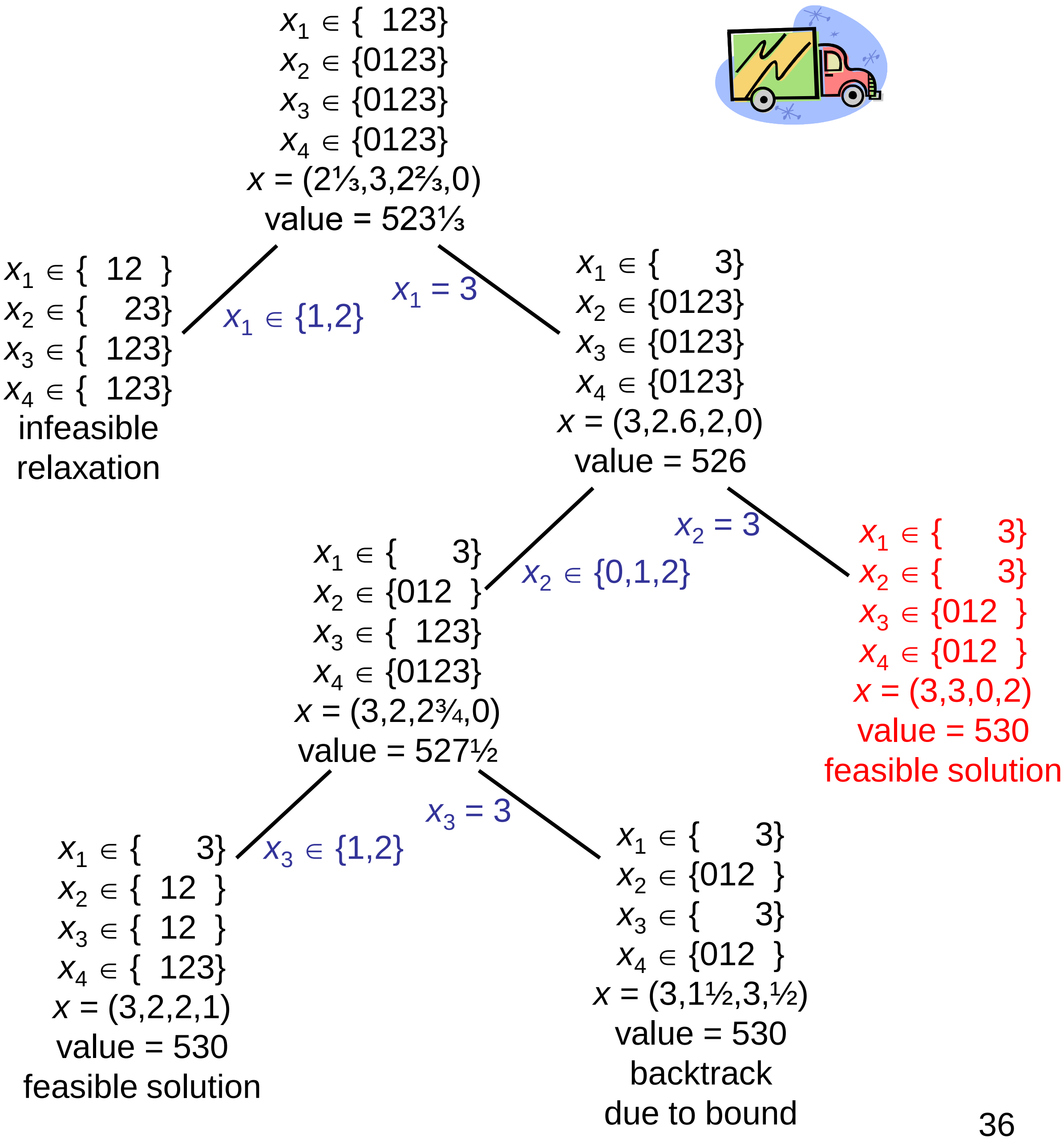
$x_3 \in \{1,2\}$

$x_1 \in \{\ \ \ 3\}$
$x_2 \in \{\ 12\ \}$
$x_3 \in \{\ 12\ \}$
$x_4 \in \{\ 123\}$
$x = (3,2,2,1)$
value = 530
feasible solution

$x_3 = 3$

$x_1 \in \{\ \ \ 3\}$
$x_2 \in \{012\ \}$
$x_3 \in \{\ \ \ 3\}$
$x_4 \in \{012\ \}$
$x = (3,1½,3,½)$
value = 530
backtrack due to bound

Branch-infer-and-relax tree



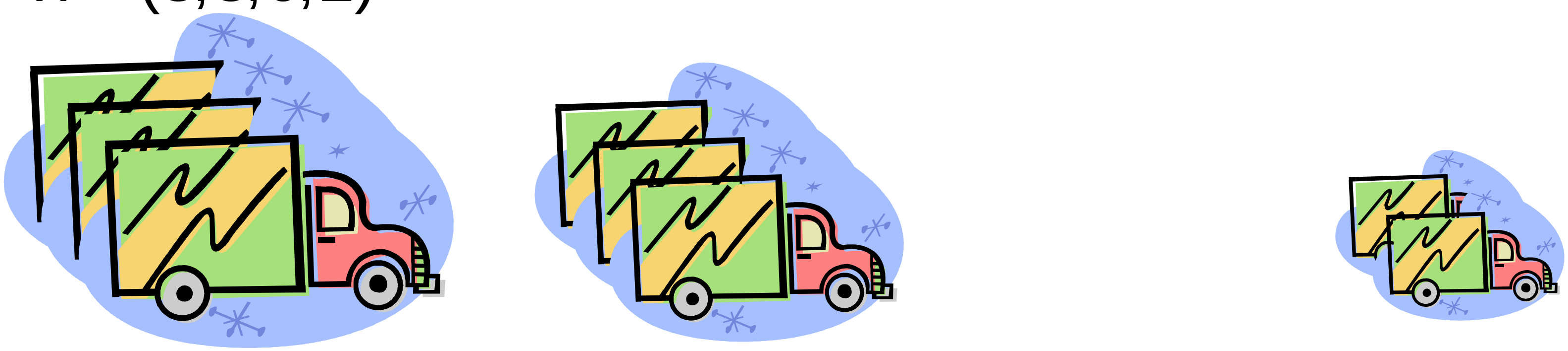
Another feasible solution found.

No better than incumbent solution, which is optimal because search has finished.

Two optimal solutions…

$x = (3, 2, 2, 1)$

$x = (3, 3, 0, 2)$



Linear Relaxation and Duality

Why Relax?

Algebraic Analysis of LP

Linear Programming Duality

LP-Based Domain Filtering

Example: Single-Vehicle Routing

Disjunctions of Linear Systems

Why Relax?

Solving a relaxation of a problem can:

- Tighten variable bounds.
- Possibly solve original problem.
- Guide the search in a promising direction.
- Filter domains using reduced costs or Lagrange multipliers.
- Prune the search tree using a bound on the optimal value.
- Provide a more global view, because a single OR relaxation can pool relaxations of several constraints.

Some OR models that can provide relaxations:

- Linear programming (LP).
- Mixed integer linear programming (MILP)
  - Can itself be relaxed as an LP.
  - LP relaxation can be strengthened with cutting planes.
- Lagrangean relaxation.
- Specialized relaxations.
  - For particular problem classes.
  - For global constraints.

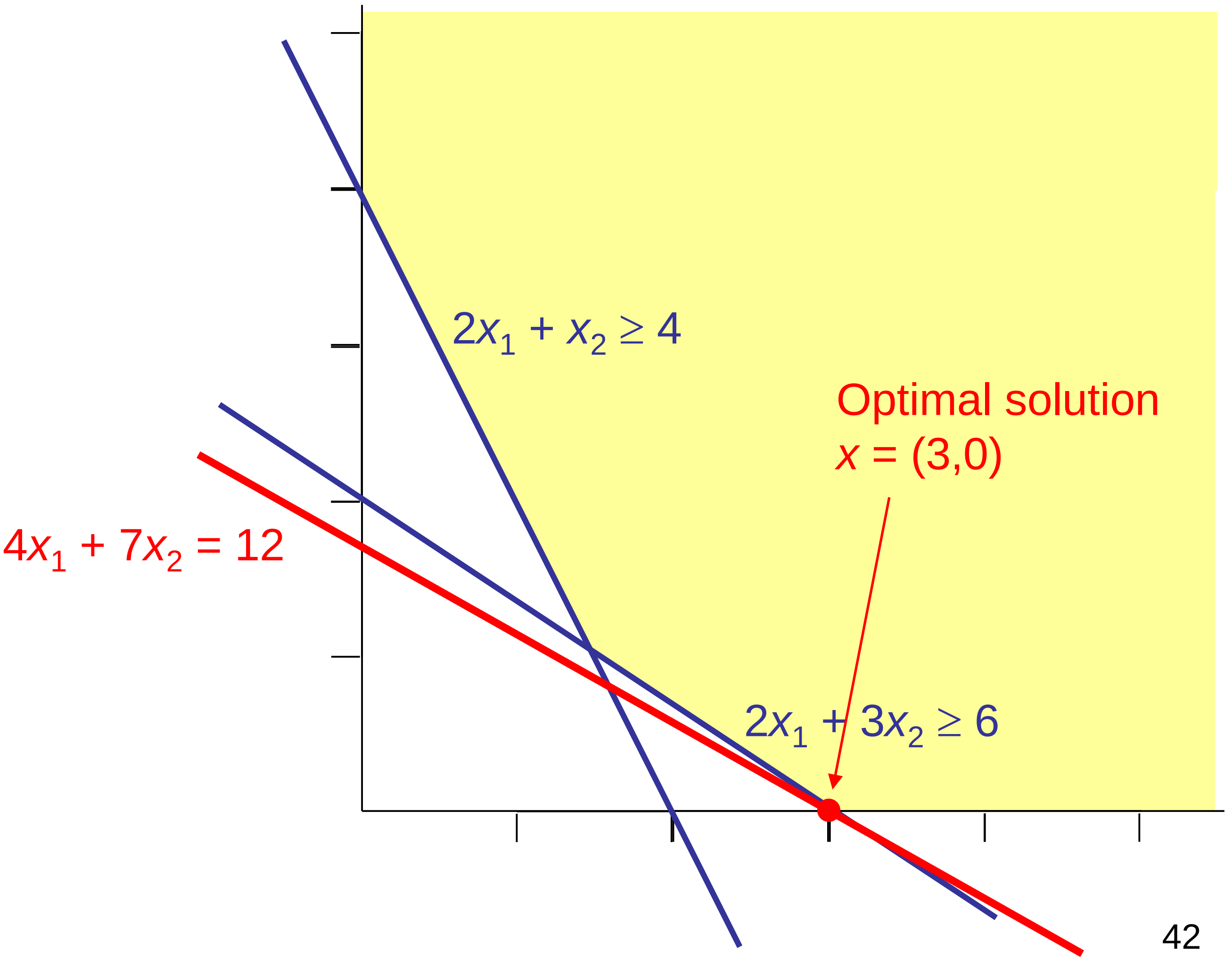
Motivation

- **Linear programming** is remarkably versatile for representing real-world problems.
- LP is by far the most widely used tool for **relaxation**.
- LP relaxations can be strengthened by **cutting planes.**
    - Based on polyhedral analysis.
- LP has an elegant and powerful **duality theory**.
    - Useful for domain filtering, and much else.
- The LP problem is **extremely well solved**.

Algebraic Analysis of LP

An example…

$$\min \ 4x_1 + 7x_2$$
$$2x_1 + 3x_2 \geq 6$$
$$2x_1 + x_2 \geq 4$$
$$x_1, x_2 \geq 0$$



Algebraic Analysis of LP

Rewrite

$$\min\ 4x_1 + 7x_2$$
$$2x_1 + 3x_2 \geq 6$$
$$2x_1 + x_2 \geq 4$$
$$x_1, x_2 \geq 0$$

as

$$\min\ 4x_1 + 7x_2$$
$$2x_1 + 3x_2 - x_3 = 6$$
$$2x_1 + x_2 - x_4 = 4$$
$$x_1, x_2, x_3, x_4 \geq 0$$

In general an LP has the form

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

Algebraic analysis of LP

Write

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

$m \times n$ matrix

as

$$\min c_B x_B + c_N x_N$$
$$B x_B + N x_N = b$$
$$x_B, x_N \geq 0$$

Basic variables

Nonbasic variables

where

$$A = [B \; N]$$

Any set of m linearly independent columns of A.

These form a **basis** for the space spanned by the columns.

Algebraic analysis of LP

Write
$$\min cx \quad Ax = b \quad x \geq 0$$

as
$$\min c_B x_B + c_N x_N \quad B x_B + N x_N = b \quad x_B, x_N \geq 0$$

where
$$A = [B\ N]$$

Solve constraint equation for x_B:
$$x_B = B^{-1}b - B^{-1}N x_N$$

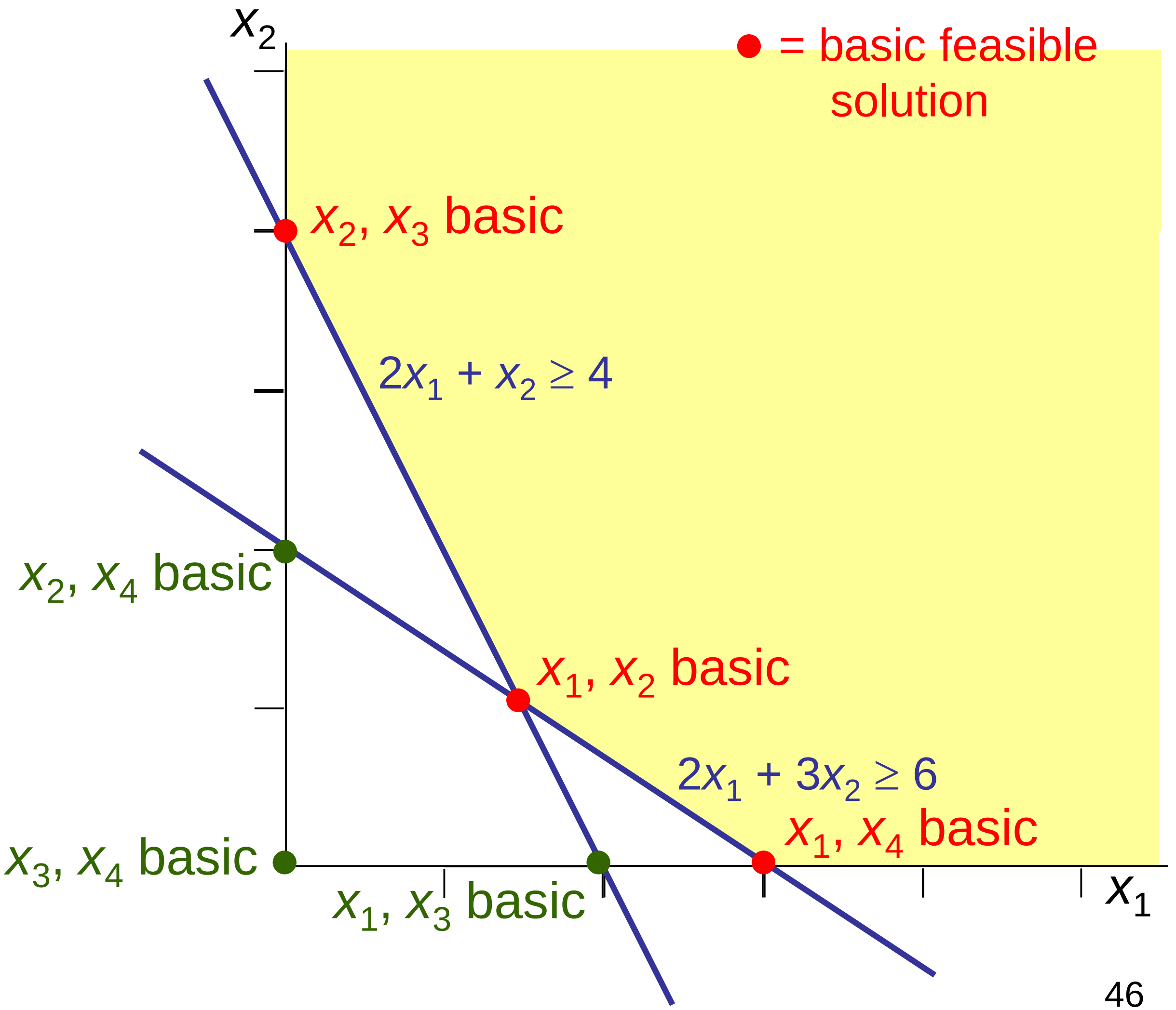
All solutions can be obtained by setting x_N to some value.

The solution is **basic** if $x_N = 0$.

It is a **basic feasible solution** if $x_N = 0$ and $x_B \geq 0$.

Example…

$$\min \ 4x_1 + 7x_2$$
$$2x_1 + 3x_2 - x_3 = 6$$
$$2x_1 + x_2 - x_4 = 4$$
$$x_1, x_2, x_3, x_4 \geq 0$$



Algebraic analysis of LP

Write $\min cx$ as $\min c_B x_B + c_N x_N$ where

$$Ax = b \qquad Bx_B + Nx_N = b \qquad A = [B\ N]$$

$$x \geq 0 \qquad x_B, x_N \geq 0$$

Solve constraint equation for x_B: $x_B = B^{-1}b - B^{-1}Nx_N$

Express cost in terms of nonbasic variables:

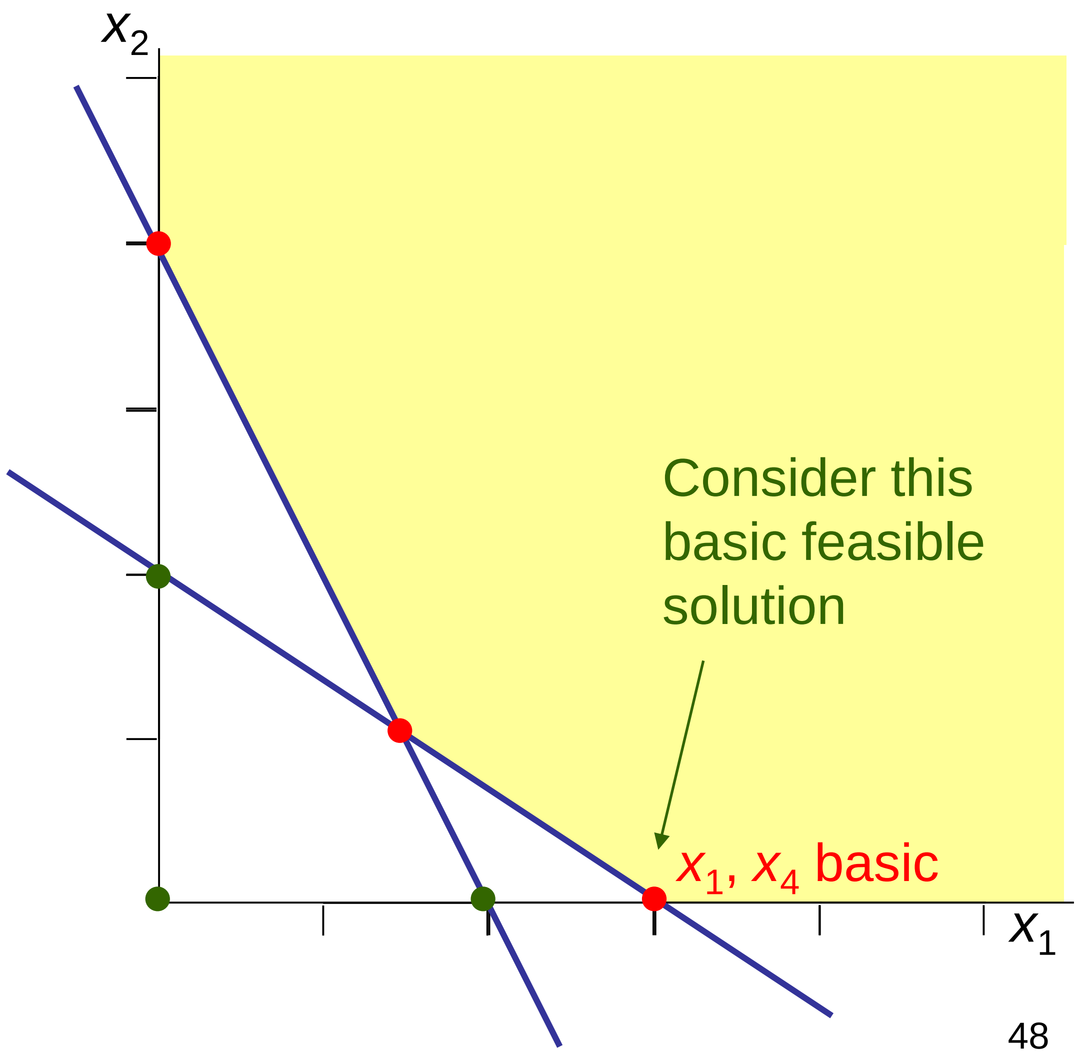
$$c_B B^{-1} b + (c_N - c_B B^{-1} N) x_N$$

Vector of reduced costs

Since $x_N \geq 0$, basic solution $(x_B,0)$ is optimal if reduced costs are nonnegative.

Example…

$$\min \; 4x_1 + 7x_2$$
$$2x_1 + 3x_2 - x_3 = 6$$
$$2x_1 + x_2 - x_4 = 4$$
$$x_1, x_2, x_3, x_4 \geq 0$$



Example…

Write…

$$\min \ 4x_1 + 7x_2$$
$$2x_1 + 3x_2 - x_3 = 6$$
$$2x_1 + x_2 - x_4 = 4$$
$$x_1, x_2, x_3, x_4 \geq 0$$

as…

$$\min \ \underbrace{[4 \ \ 0]\begin{bmatrix} x_1 \\ x_4 \end{bmatrix}}_{c_B x_B} + \underbrace{[7 \ \ 0]\begin{bmatrix} x_2 \\ x_3 \end{bmatrix}}_{c_N x_N}$$

$$\underbrace{\begin{bmatrix} 2 & 0 \\ 2 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_4 \end{bmatrix}}_{Bx_B} + \underbrace{\begin{bmatrix} 3 & -1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_4 \end{bmatrix}}_{Nx_N} = \underbrace{\begin{bmatrix} 6 \\ 4 \end{bmatrix}}_{b}$$

$$\begin{bmatrix} x_1 \\ x_4 \end{bmatrix}, \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Example…

$$\min \underbrace{\begin{bmatrix}4 & 0\end{bmatrix}\begin{bmatrix}x_1\\x_4\end{bmatrix}}_{c_B x_B} + \underbrace{\begin{bmatrix}7 & 0\end{bmatrix}\begin{bmatrix}x_2\\x_3\end{bmatrix}}_{c_N x_N}$$

$$\underbrace{\begin{bmatrix}2 & 0\\2 & -1\end{bmatrix}\begin{bmatrix}x_1\\x_4\end{bmatrix}}_{Bx_B} + \underbrace{\begin{bmatrix}3 & -1\\1 & 0\end{bmatrix}\begin{bmatrix}x_1\\x_4\end{bmatrix}}_{Nx_N} = \underbrace{\begin{bmatrix}6\\4\end{bmatrix}}_{b}$$

$$\begin{bmatrix}x_1\\x_4\end{bmatrix}, \begin{bmatrix}x_1\\x_4\end{bmatrix} \geq \begin{bmatrix}0\\0\end{bmatrix}$$

Example…

$$\min \underbrace{[4\ \ 0]\begin{bmatrix} x_1 \\ x_4 \end{bmatrix}}_{c_B x_B} + \underbrace{[7\ \ 0]\begin{bmatrix} x_2 \\ x_3 \end{bmatrix}}_{c_N x_N}$$

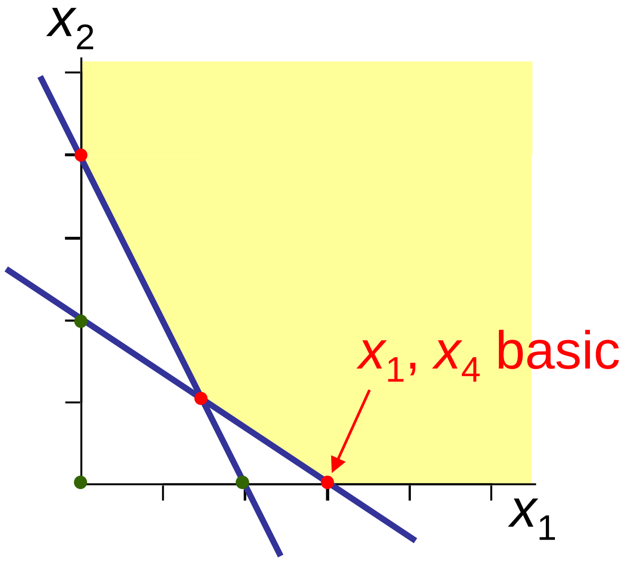
$$\underbrace{\begin{bmatrix} 2 & 0 \\ 2 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_4 \end{bmatrix}}_{Bx_B} + \underbrace{\begin{bmatrix} 3 & -1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_4 \end{bmatrix}}_{Nx_N} = \underbrace{\begin{bmatrix} 6 \\ 4 \end{bmatrix}}_{b}$$

$$\begin{bmatrix} x_1 \\ x_4 \end{bmatrix}, \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Basic solution is

$$x_B = B^{-1}b - B^{-1}Nx_N = B^{-1}b$$

$$= \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1/2 & 0 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 6 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$



Example…

$$\min \underbrace{\begin{bmatrix}4 & 0\end{bmatrix}\begin{bmatrix}x_1\\x_4\end{bmatrix}}_{c_B x_B} + \underbrace{\begin{bmatrix}7 & 0\end{bmatrix}\begin{bmatrix}x_2\\x_3\end{bmatrix}}_{c_N x_N}$$

$$\underbrace{\begin{bmatrix}2 & 0\\2 & -1\end{bmatrix}\begin{bmatrix}x_1\\x_4\end{bmatrix}}_{Bx_B} + \underbrace{\begin{bmatrix}3 & -1\\1 & 0\end{bmatrix}\begin{bmatrix}x_1\\x_4\end{bmatrix}}_{Nx_N} = \begin{bmatrix}6\\4\end{bmatrix}$$

$$\begin{bmatrix}x_1\\x_4\end{bmatrix}, \begin{bmatrix}x_1\\x_4\end{bmatrix} \geq \begin{bmatrix}0\\0\end{bmatrix}$$

Basic solution is

$$x_B = B^{-1}b - B^{-1}Nx_N = B^{-1}b$$

$$= \begin{bmatrix}x_1\\x_4\end{bmatrix} = \begin{bmatrix}1/2 & 0\\1 & -1\end{bmatrix}\begin{bmatrix}6\\4\end{bmatrix} = \begin{bmatrix}3\\2\end{bmatrix}$$

Reduced costs are

$$c_N - c_B B^{-1}N$$

$$= \begin{bmatrix}7 & 0\end{bmatrix} - \begin{bmatrix}4 & 0\end{bmatrix}\begin{bmatrix}1/2 & 0\\1 & -1\end{bmatrix}\begin{bmatrix}3 & -1\\1 & 0\end{bmatrix}$$

$$= \begin{bmatrix}1 & 2\end{bmatrix} \geq \begin{bmatrix}0 & 0\end{bmatrix}$$

Solution is optimal

Linear Programming Duality

An LP can be viewed as an inference problem…

$$\min\ cx \quad = \quad \max\ v$$

$$Ax \geq b \qquad\qquad Ax \geq b \overset{x\geq 0}{\Rightarrow} cx \geq v$$

$$x \geq 0$$

implies

Dual problem: Find the tightest lower bound on the objective function that is implied by the constraints.

An LP can be viewed as an inference problem…

$$\min\ cx \quad = \quad \max\ v$$

$$Ax \geq b \qquad\qquad Ax \geq b \overset{x\geq 0}{\Rightarrow} cx \geq v$$

$$x \geq 0$$

That is, some **surrogate** (nonnegative linear combination) of $Ax \geq b$ dominates $cx \geq v$

From Farkas Lemma: If $Ax \geq b$, $x \geq 0$ is feasible,

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

for some $\lambda \geq 0$

$\lambda A \leq c$ and $\lambda b \geq v$

An LP can be viewed as an inference problem…

$$\min\ cx \quad = \quad \max\ v \quad = \quad \max\ \lambda b$$

$$Ax \ge b \qquad Ax \ge b \overset{x\ge 0}{\Rightarrow} cx \ge v \qquad \lambda A \le c$$

$$x \ge 0 \qquad \lambda \ge 0$$

This is the **classical LP dual**

From Farkas Lemma: If $Ax \ge b$, $x \ge 0$ is feasible,

$Ax \ge b \overset{x\ge 0}{\Rightarrow} cx \ge v$ iff $\lambda Ax \ge \lambda b$ dominates $cx \ge v$ for some $\lambda \ge 0$

$\lambda A \le c$ and $\lambda b \ge v$

This equality is called **strong duality.**

$$\min \; cx \quad = \quad \max \; \lambda b$$
$$Ax \geq b \qquad \lambda A \leq c$$
$$x \geq 0 \qquad \lambda \geq 0$$

This is the **classical LP dual**

If $Ax \geq b$, $x \geq 0$ is feasible

Note that the dual of the dual is the **primal** (i.e., the original LP).

Example

Primal

$$\min\ 4x_1 + 7x_2 \quad =$$

$$2x_1 + 3x_2 \geq 6 \quad (\lambda_1)$$

$$2x_1 + x_2 \geq 4 \quad (\lambda_1)$$

$$x_1, x_2 \geq 0$$

Dual

$$\max\ 6\lambda_1 + 4\lambda_2 \quad = 12$$

$$2\lambda_1 + 2\lambda_2 \leq 4 \quad (x_1)$$

$$3\lambda_1 + \lambda_2 \leq 7 \quad (x_2)$$

$$\lambda_1, \lambda_2 \geq 0$$

A dual solution is $(\lambda_1, \lambda_2) = (2,0)$

$$2x_1 + 3x_2 \geq 6 \quad \cdot(\lambda_1 = 2)$$

$$2x_1 + x_2 \geq 4 \quad \cdot(\lambda_2 = 0)$$

← Dual multipliers

$$4x_1 + 6x_2 \geq 12$$ ← Surrogate

↓ dominates

$$4x_1 + 7x_2 \geq 12$$ ← Tightest bound on cost

Weak Duality

If x* is feasible in the primal problem

$\min \; cx$

$Ax \geq b$

$x \geq 0$

and λ* is feasible in the dual problem

$\max \; \lambda b$

$\lambda A \leq c$

$\lambda \geq 0$

then $cx^* \geq \lambda^* b$.

This is because

$cx^* \geq \lambda^* A x^* \geq \lambda^* b$

λ^* is dual feasible and $x^* \geq 0$

x^* is primal feasible and $\lambda^* \geq 0$

Dual multipliers as marginal costs

Suppose we perturb the RHS of an LP (i.e., change the requirement levels):

$$\min \; cx$$
$$Ax \geq b + \Delta b$$
$$x \geq 0$$

The dual of the perturbed LP has the same constraints at the original LP:

$$\max \; \lambda(b + \Delta b)$$
$$\lambda A \leq c$$
$$\lambda \geq 0$$

So an optimal solution λ^* of the original dual is feasible in the perturbed dual.

Dual multipliers as marginal costs

Suppose we perturb the RHS of an LP (i.e., change the requirement levels):

$$\min \; cx$$
$$Ax \geq b + \Delta b$$
$$x \geq 0$$

By weak duality, the optimal value of the perturbed LP is at least $\lambda^*(b + \Delta b) = \boxed{\lambda^* b} + \lambda^* \Delta b$.

Optimal value of original LP, by strong duality.

So λ_i^* is a lower bound on the marginal cost of increasing the i-th requirement by one unit ($\Delta b_i = 1$).

If $\lambda_i^* > 0$, the i-th constraint must be tight **(complementary slackness).**

Dual of an LP in equality form

Primal

$$\min c_B x_B + c_N x_N$$

$$Bx_B + Nx_N = b \qquad (\lambda)$$

$$x_B, x_N \geq 0$$

Dual

$$\max \lambda b$$

$$\lambda B \leq c_B \qquad (x_B)$$

$$\lambda N \leq c_N \qquad (x_B)$$

$$\lambda \text{ unrestricted}$$

Dual of an LP in equality form

Primal

$$\min c_B x_B + c_N x_N$$

$$Bx_B + Nx_N = b \qquad (\lambda)$$

$$x_B, x_N \geq 0$$

Dual

$$\max \lambda b$$

$$\lambda B \leq c_B \qquad (x_B)$$

$$\lambda N \leq c_N \qquad (x_B)$$

$$\lambda \text{ unrestricted}$$

Recall that reduced cost vector is $c_N - \boxed{c_B B^{-1}} N = c_N - \lambda N$

λ

this solves the dual
if (x_B,0) solves the primal

Dual of an LP in equality form

Primal

$$\min c_B x_B + c_N x_N$$
$$B x_B + N x_N = b \qquad (\lambda)$$
$$x_B, x_N \geq 0$$

Dual

$$\max \lambda b$$
$$\lambda B \leq c_B \qquad (x_B)$$
$$\lambda N \leq c_N \qquad (x_B)$$
$$\lambda \text{ unrestricted}$$

Recall that reduced cost vector is $c_N - \boxed{c_B B^{-1}} N = c_N - \lambda N$

λ ← this solves the dual if $(x_B,0)$ solves the primal

Check:
$$\lambda B = c_B B^{-1} B = c_B$$
$$\lambda N = c_B B^{-1} N \leq c_N$$

Because reduced cost is nonnegative at optimal solution $(x_B,0)$.

Dual of an LP in equality form

Primal

$$\min c_B x_B + c_N x_N$$
$$Bx_B + Nx_N = b \qquad (\lambda)$$
$$x_B, x_N \geq 0$$

Dual

$$\max \lambda b$$
$$\lambda B \leq c_B \qquad (x_B)$$
$$\lambda N \leq c_N \qquad (x_B)$$
$$\lambda \text{ unrestricted}$$

Recall that reduced cost vector is $c_N - c_B B^{-1} N = c_N - \lambda N$

$c_B B^{-1} = \lambda$ ← this solves the dual if $(x_B,0)$ solves the primal

In the example,

$$\lambda = c_B B^{-1} = [4 \quad 0]\begin{bmatrix} 1/2 & 0 \\ 1 & -1 \end{bmatrix} = [2 \quad 0]$$

Dual of an LP in equality form

Primal

$$\min c_B x_B + c_N x_N$$

$$B x_B + N x_N = b \qquad (\lambda)$$

$$x_B, x_N \geq 0$$

Dual

$$\max \lambda b$$

$$\lambda B \leq c_B \qquad (x_B)$$

$$\lambda N \leq c_N \qquad (x_B)$$

λ unrestricted

Recall that reduced cost vector is $c_N - \underbrace{c_B B^{-1}}_{\lambda} N = c_N - \lambda N$

Note that the reduced cost of an individual variable x_j is $r_j = c_j - \lambda A_j$

Column j of A

LP-based Domain Filtering

Let $\begin{array}{l} \min\ cx \\ Ax \geq b \\ x \geq 0 \end{array}$ be an LP relaxation of a CP problem.

- One way to filter the domain of x_j is to minimize and maximize x_j subject to $Ax \geq b$, $x \geq 0$.
  - This is time consuming.
- A faster method is to use **dual multipliers** to derive valid inequalities.
  - A special case of this method uses **reduced costs** to bound or fix variables.
  - **Reduced-cost variable fixing** is a widely used technique in OR.

Suppose:

$$\begin{aligned} \min\ & cx \\ & Ax \geq b \\ & x \geq 0 \end{aligned}$$

has optimal solution x^*, optimal value v^*, and optimal dual solution λ^*.

…and $\lambda_i^* > 0$, which means the i-th constraint is tight (complementary slackness);

…and the LP is a relaxation of a CP problem;

…and we have a feasible solution of the CP problem with value U, so that U is an upper bound on the optimal value.

Supposing $\begin{array}{l} \min \ cx \\ Ax \geq b \\ x \geq 0 \end{array}$ has optimal solution x^*, optimal value v^*, and optimal dual solution λ^*:

If x were to change to a value other than x^*, the LHS of i-th constraint $A^i x \geq b_i$ would change by some amount Δb_i.

Since the constraint is tight, this would increase the optimal value as much as changing the constraint to $A^i x \geq b_i + \Delta b_i$.

So it would increase the optimal value at least $\lambda_i^* \Delta b_i$.

Supposing $\min \; cx$, $Ax \geq b$, $x \geq 0$ has optimal solution x^*, optimal value v^*, and optimal dual solution λ^*:

We have found: a change in x that changes $A^i x$ by Δb_i increases the optimal value of LP at least $\lambda_i^* \Delta b_i$.

Since optimal value of the LP ≤ optimal value of the CP ≤ U, we have $\lambda_i^* \Delta b_i \leq U - v^*$, or

$$\Delta b_i \leq \frac{U - v^*}{\lambda_i^*}$$

Supposing $\begin{array}{l} \min\ cx \\ Ax \geq b \\ x \geq 0 \end{array}$ has optimal solution x^*, optimal value v^*, and optimal dual solution λ^*:

We have found: a change in x that changes $A^i x$ by Δb_i increases the optimal value of LP at least $\lambda_i^* \Delta b_i$.

Since optimal value of the LP $\leq$ optimal value of the CP $\leq U$,
we have $\lambda_i^* \Delta b_i \leq U - v^*$, or

$$\Delta b_i \leq \frac{U - v^*}{\lambda_i^*}$$

Since $\Delta b_i = A^i x - A^i x^* = A^i x - b_i$, this implies the inequality

$$A^i x \leq b_i + \frac{U - v^*}{\lambda_i^*}$$

…which can be propagated.

Example

$$\min\ 4x_1 + 7x_2$$
$$2x_1 + 3x_2 \geq 6 \qquad (\lambda_1 = 2)$$
$$2x_1 + x_2 \geq 4 \qquad (\lambda_1 = 0)$$
$$x_1, x_2 \geq 0$$

Suppose we have a feasible solution of the original CP with value U = 13.

Since the first constraint is tight, we can propagate the inequality

$$A^1 x \leq b_1 + \frac{U - v^*}{\lambda_1^*}$$

or $$2x_1 + 3x_2 \leq 6 + \frac{13 - 12}{2} = 6.5$$

Reduced-cost domain filtering

Suppose $x_j^* = 0$, which means the constraint $x_j \geq 0$ is tight.

The inequality $A^i x \leq b_i + \dfrac{U - v^*}{\lambda_i^*}$ becomes $x_j \leq \dfrac{U - v^*}{\boxed{r_j}}$

The dual multiplier for $x_j \geq 0$ is the reduced cost r_j of x_j, because increasing x_j (currently 0) by 1 increases optimal cost by r_j.

Similar reasoning can bound a variable below when it is at its upper bound.

Example

$$\min\ 4x_1 + 7x_2$$
$$2x_1 + 3x_2 \geq 6 \qquad (\lambda_1 = 2)$$
$$2x_1 + x_2 \geq 4 \qquad (\lambda_1 = 0)$$
$$x_1, x_2 \geq 0$$

Suppose we have a feasible solution of the original CP with value U = 13.

Since x_2^* = 0, we have $x_2 \leq \dfrac{U - v^*}{r_2}$

or $x_2 \leq \dfrac{13-12}{2} = 0.5$

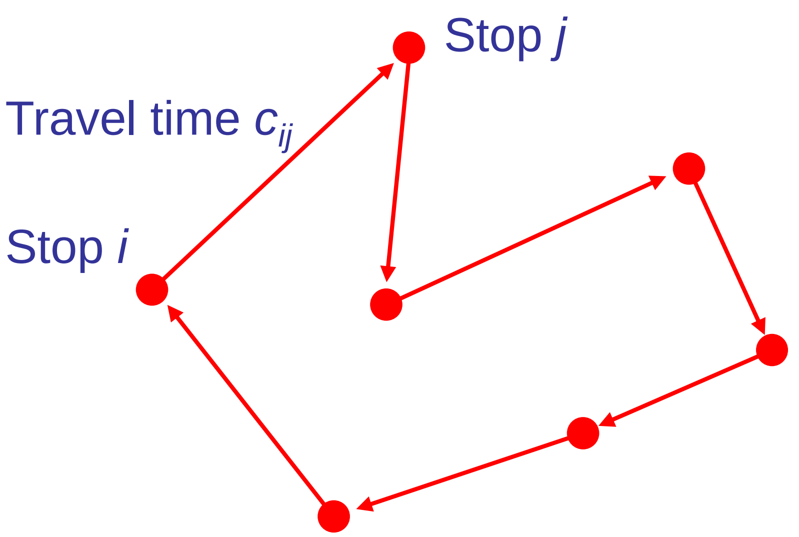
If x_2 is required to be integer, we can fix it to zero. This is **reduced-cost variable fixing.**

Example: Single-Vehicle Routing

A vehicle must make several stops and return home, perhaps subject to time windows.

The objective is to find the order of stops that minimizes travel time.

This is also known as the **traveling salesman problem** (with time windows).



Assignment Relaxation

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

= 1 if stop *i* immediately precedes stop *j*

$$\sum_j x_{ij} = \sum_j x_{ji} = 1, \text{ all } i$$

Stop *i* is preceded and followed by exactly one stop.

$$x_{ij} \in \{0,1\}, \text{ all } i, j$$

Assignment Relaxation

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

= 1 if stop *i* immediately precedes stop *j*

$$\sum_j x_{ij} = \sum_j x_{ji} = 1, \text{ all } i$$

Stop *i* is preceded and followed by exactly one stop.

$$0 \le x_{ij} \le 1, \text{ all } i, j$$

Because this problem is **totally unimodular**, it can be solved as an LP.

The relaxation provides a very weak lower bound on the optimal value.

But **reduced-cost variable fixing** can be very useful in a CP context.

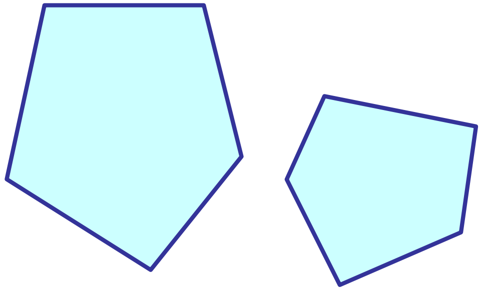
Disjunctions of linear systems

Disjunctions of linear systems often occur naturally in problems and can be given a convex hull relaxation.

A disjunction of linear systems represents a union of polyhedra.

$$\min \ cx$$

$$\bigvee_k \left(A^k x \geq b^k \right)$$



Relaxing a disjunction of linear systems

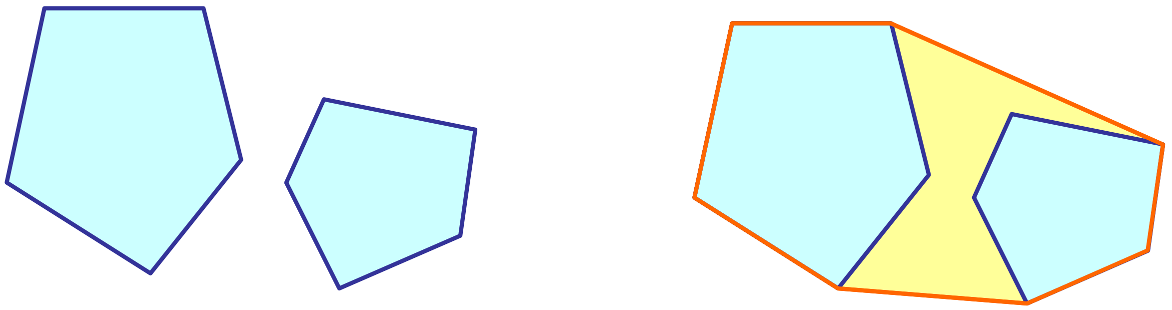
Disjunctions of linear systems often occur naturally in problems and can be given a convex hull relaxation.

A disjunction of linear systems represents a union of polyhedra.

We want a convex hull relaxation (tightest linear relaxation).

$$\min \ cx$$

$$\bigvee_k \left(A^k x \geq b^k \right)$$



Relaxing a disjunction of linear systems

Disjunctions of linear systems often occur naturally in problems and can be given a convex hull relaxation.

The closure of the convex hull of

$$\min \; cx$$
$$\bigvee_k \left(A^k x \geq b^k \right)$$

…is described by

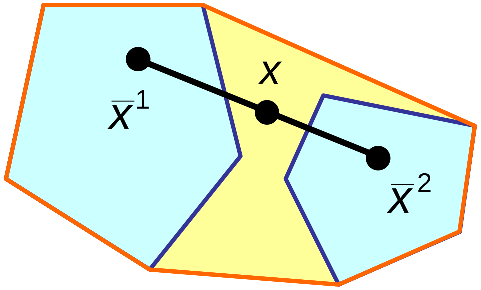
$$\min \; cx$$
$$A^k x^k \geq b^k y_k, \text{ all } k$$
$$\sum_k y_k = 1$$
$$x = \sum_k x^k$$
$$0 \leq y_k \leq 1$$

Why?

To derive convex hull relaxation of a disjunction…

$$\min\ cx$$
$$A^k \bar{x}^k \ge b^k,\ \text{all } k$$
$$\sum_k y_k = 1$$
$$x = \sum_k y_k \bar{x}^k$$
$$0 \le y_k \le 1$$

Write each solution as a convex combination of points in the polyhedron



Convex hull relaxation (tightest linear relaxation)

Why?

To derive convex hull relaxation of a disjunction…

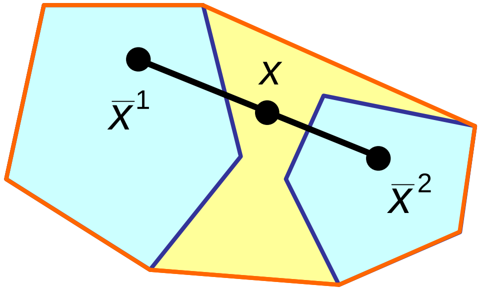
Write each solution as a convex combination of points in the polyhedron

$$\begin{aligned} &\min\ cx \\ &A^k \bar{x}^k \geq b^k,\ \text{all } k \\ &\sum_k y_k = 1 \\ &x = \sum_k y_k \bar{x}^k \\ &0 \leq y_k \leq 1 \end{aligned}$$

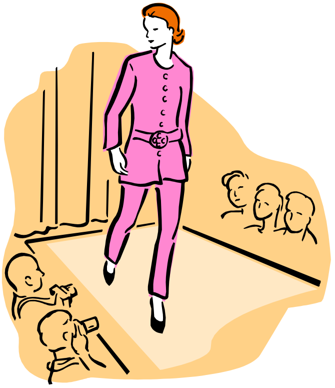
Change of variable

$$x^k = y_k \bar{x}^k$$

$$\begin{aligned} &\min\ cx \\ &A^k x^k \geq b^k y_k,\ \text{all } k \\ &\sum_k y_k = 1 \\ &x = \sum_k x^k \\ &0 \leq y_k \leq 1 \end{aligned}$$



Convex hull relaxation (tightest linear relaxation)



Mixed Integer/Linear Modeling

MILP Representability

Disjunctive Modeling

Knapsack Modeling

Motivation

A **mixed integer/linear programming** (MILP) problem has the form

$$\begin{aligned} &\min\ cx + dy \\ &Ax + by \geq b \\ &x, y \geq 0 \\ &y \text{ integer} \end{aligned}$$

- We can **relax** a CP problem by modeling some constraints with an MILP.
- If desired, we can then **relax the MILP** by dropping the integrality constraint, to obtain an LP.
- The LP relaxation can be strengthened with **cutting planes**.
- The first step is to learn **how to write** MILP models.

MILP Representability

A subset S of $\mathbb{R}^n$ is **MILP representable** if it is the projection onto x of some MILP constraint set of the form

$$Ax + Bu + Dy \geq b$$

$$x, y \geq 0$$

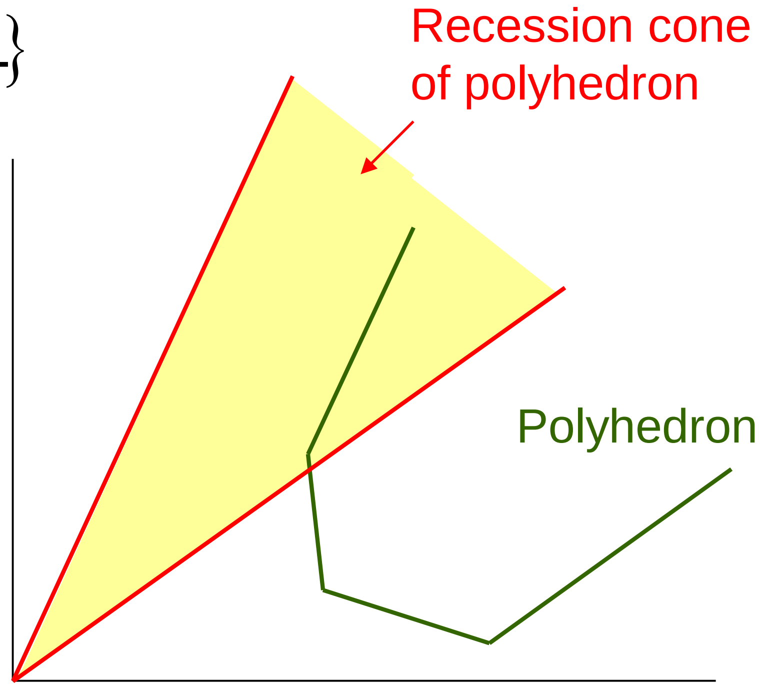
$$x \in \mathbb{R}^n,\ \ u \in \mathbb{R}^m,\ \ y_k \in \{0,1\}$$

MILP Representability

A subset S of $\mathbb{R}^n$ is **MILP representable** if it is the projection onto x of some MILP constraint set of the form

$$Ax + Bu + Dy \geq b$$
$$x, y \geq 0$$
$$x \in \mathbb{R}^n, \; u \in \mathbb{R}^m, \; y_k \in \{0,1\}$$

Theorem. $S \subset \mathbb{R}^n$ is MILP representable if and only if S is the union of finitely many polyhedra having the same recession cone.



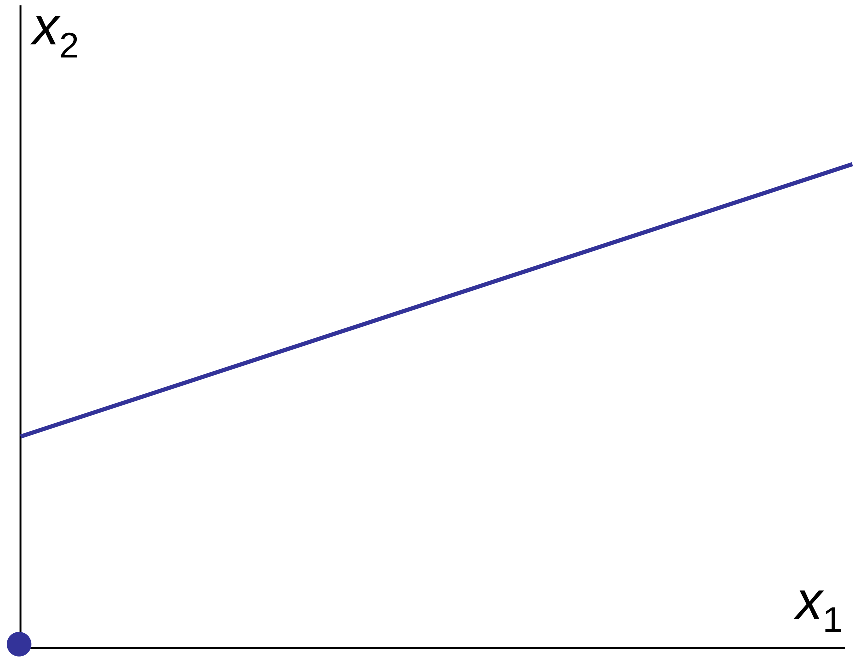
Example: Fixed charge function

Minimize a fixed charge function:

$$\min \ x_2$$

$$x_2 \geq \begin{Bmatrix} 0 & \text{if } x_1 = 0 \\ f + cx_1 & \text{if } x_1 > 0 \end{Bmatrix}$$

$$x_1 \geq 0$$



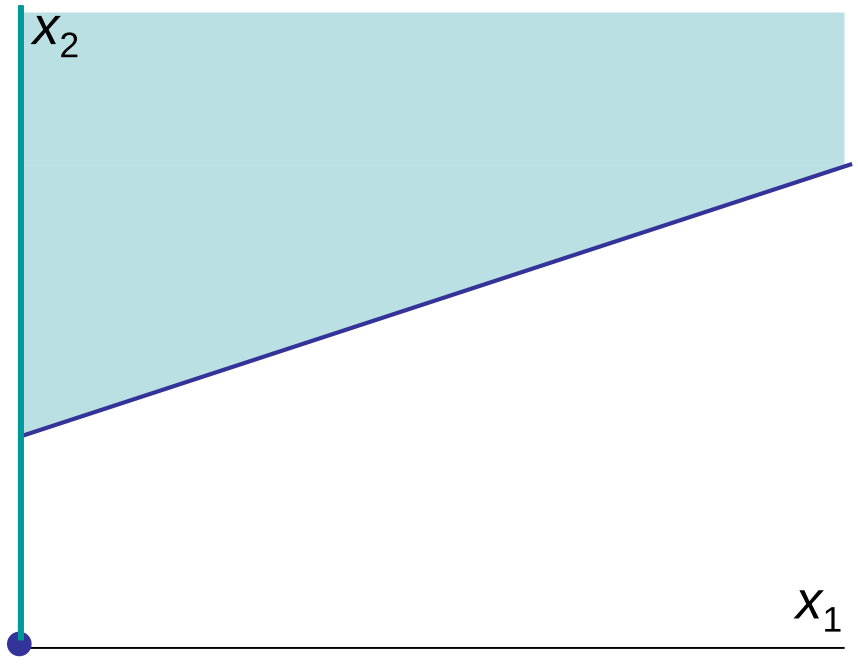
Example

Minimize a fixed charge function:

$$\min \ x_2$$

$$x_2 \geq \begin{Bmatrix} 0 & \text{if } x_1 = 0 \\ f + cx_1 & \text{if } x_1 > 0 \end{Bmatrix}$$

$$x_1 \geq 0$$



Feasible set (epigraph of the optimization problem)

Example

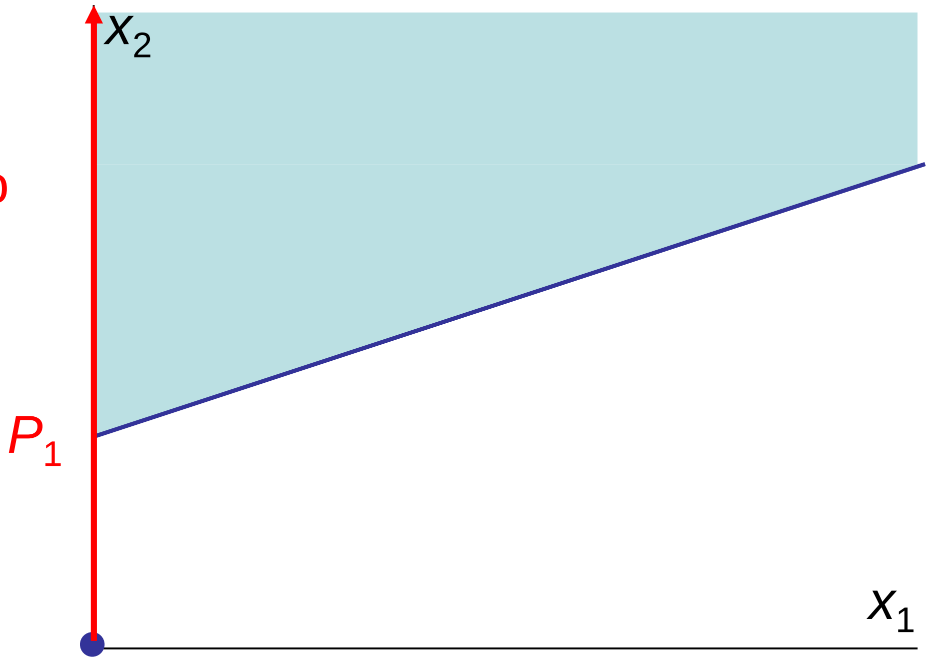
Minimize a fixed charge function:

$$\min \; x_2$$

$$x_2 \ge \begin{cases} 0 & \text{if } x_1 = 0 \\ f + cx_1 & \text{if } x_1 > 0 \end{cases}$$

$$x_1 \ge 0$$

Union of two polyhedra P_1, P_2



Example

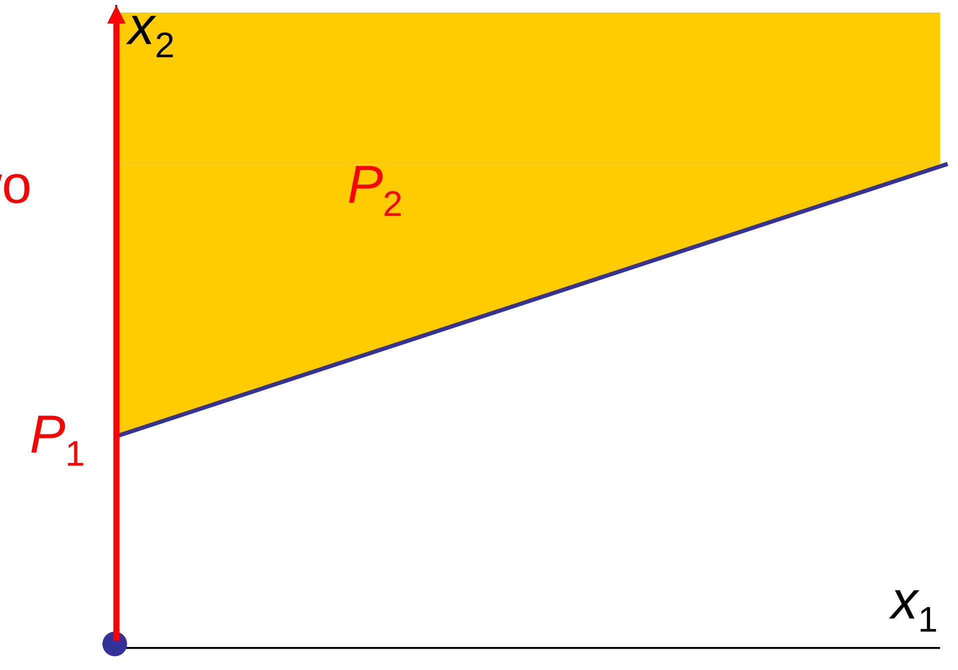
Minimize a fixed charge function:

$$\min \; x_2$$

$$x_2 \geq \begin{Bmatrix} 0 & \text{if } x_1 = 0 \\ f + cx_1 & \text{if } x_1 > 0 \end{Bmatrix}$$

$$x_1 \geq 0$$

Union of two polyhedra P_1, P_2



Example

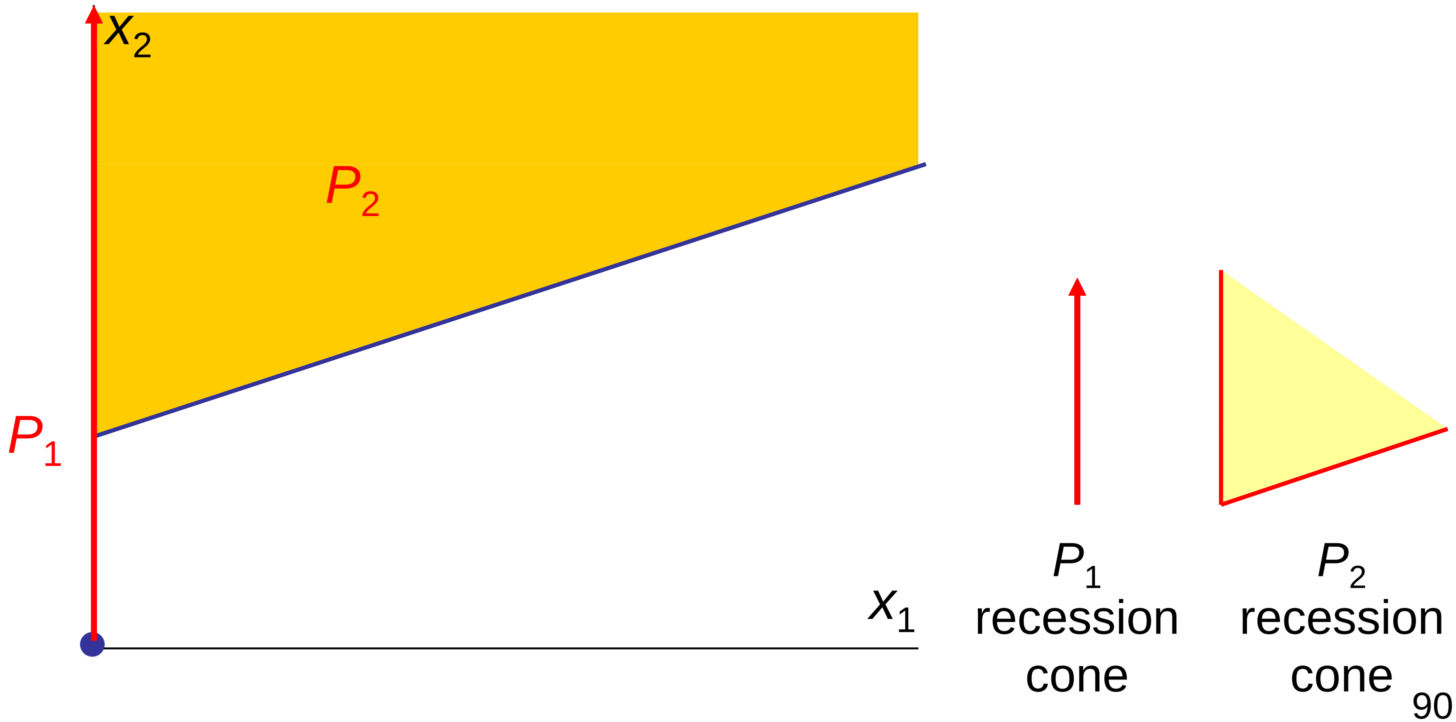
Minimize a fixed charge function:

$$\min \; x_2$$

$$x_2 \geq \begin{Bmatrix} 0 & \text{if } x_1 = 0 \\ f + cx_1 & \text{if } x_1 > 0 \end{Bmatrix}$$

$$x_1 \geq 0$$

The polyhedra have different recession cones.



Example

Minimize a fixed charge function:

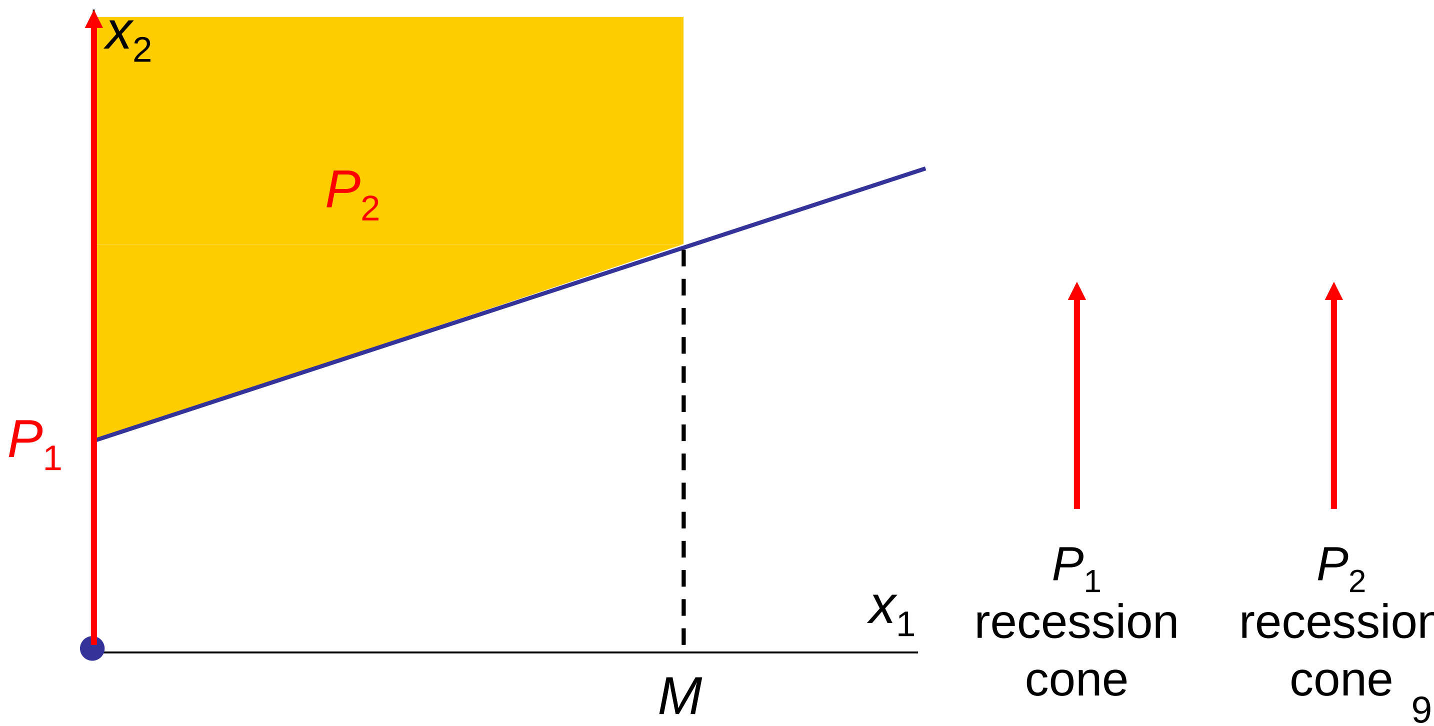
$$\min \ x_2$$

$$x_2 \geq \begin{Bmatrix} 0 & \text{if } x_1 = 0 \\ f + cx_1 & \text{if } x_1 > 0 \end{Bmatrix}$$

Add an upper bound on x_1

$$0 \leq x_1 \leq M$$

The polyhedra have the same recession cone.



Modeling a union of polyhedra

Start with a disjunction of linear systems to represent the union of polyhedra.

The kth polyhedron is $\{x \mid A^k x \geq b\}$

$$\min \; cx$$
$$\bigvee_k \left(A^k x \geq b^k \right)$$

Introduce a 0-1 variable y_k that is 1 when x is in polyhedron $\underline{k}$.

Disaggregate x to create an x^k for each k.

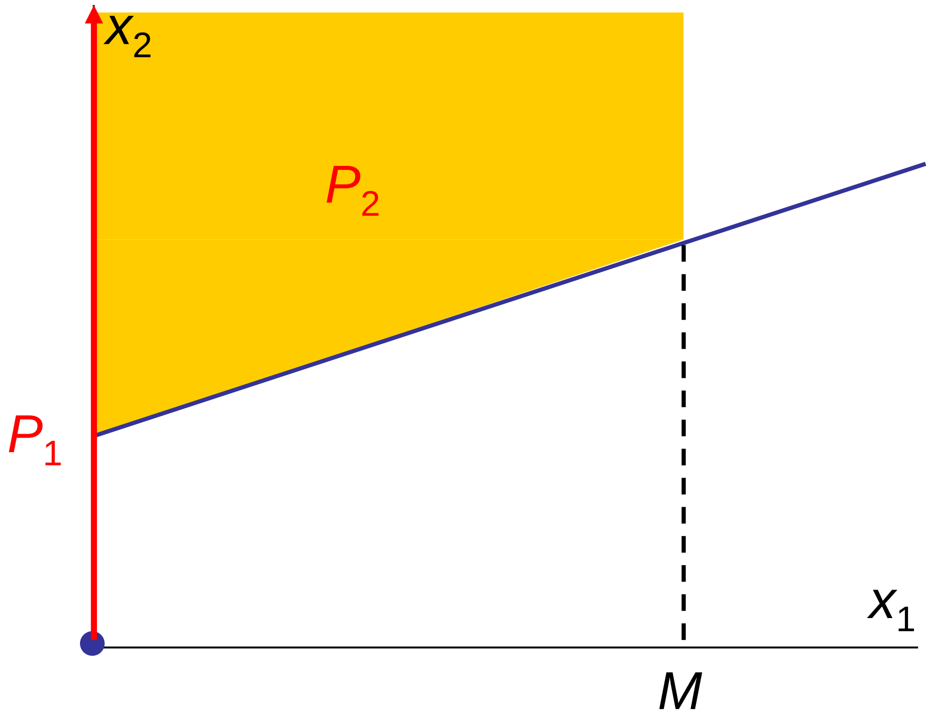
$$\min \; cx$$
$$A^k x^k \geq b^k y_k, \text{ all } k$$
$$\sum_k y_k = 1$$
$$x = \sum_k x^k$$
$$y_k \in \{0,1\}$$

Example

Start with a disjunction of linear systems to represent the union of polyhedra

$$\min \; x_2$$

$$\begin{pmatrix} x_1 = 0 \\ x_2 \ge 0 \end{pmatrix} \vee \begin{pmatrix} 0 \le x_1 \le M \\ x_2 \ge f + cx_1 \end{pmatrix}$$



Example

Start with a disjunction of linear systems to represent the union of polyhedra

$$\min \; x_2$$
$$\begin{pmatrix} x_1 = 0 \\ x_2 \ge 0 \end{pmatrix} \vee \begin{pmatrix} 0 \le x_1 \le M \\ x_2 \ge f + cx_1 \end{pmatrix}$$

Introduce a 0-1 variable y_k that is 1 when x is in polyhedron <u>k</u>.

Disaggregate x to create an x^k for each k.

$$\min \; cx$$
$$x_1^1 = 0, \;\; x_2^1 \ge 0$$
$$0 \le x_1^2 \le My_2, \quad -cx_1^2 + x_2^2 \ge fy_2$$
$$y_1 + y_2 = 1, \;\; y_k \in \{0,1\}$$
$$x = x^1 + x^2$$

Example

To simplify:

Replace x_1^2 with x_1.

Replace x_2^2 with x_2.

Replace y_2 with y.

$$\min \; x_2$$
$$x_1^1 = 0, \;\; x_2^1 \geq 0$$
$$0 \leq x_1^2 \leq My_2, \quad -cx_1^2 + x_2^2 \geq fy_2$$
$$y_1 + y_2 = 1, \;\; y_k \in \{0,1\}$$
$$x = x^1 + x^2$$

This yields

$$\min \; x_2$$
$$0 \leq x_1 \leq My$$
$$x_2 \geq fy + cx_1$$
$$y \in \{0,1\}$$

or

$$\min \; fy + cx$$
$$0 \leq x \leq My$$
$$y \in \{0,1\}$$

"Big *M*"

Disjunctive Modeling

Disjunctions often occur naturally in problems and can be given an MILP model.

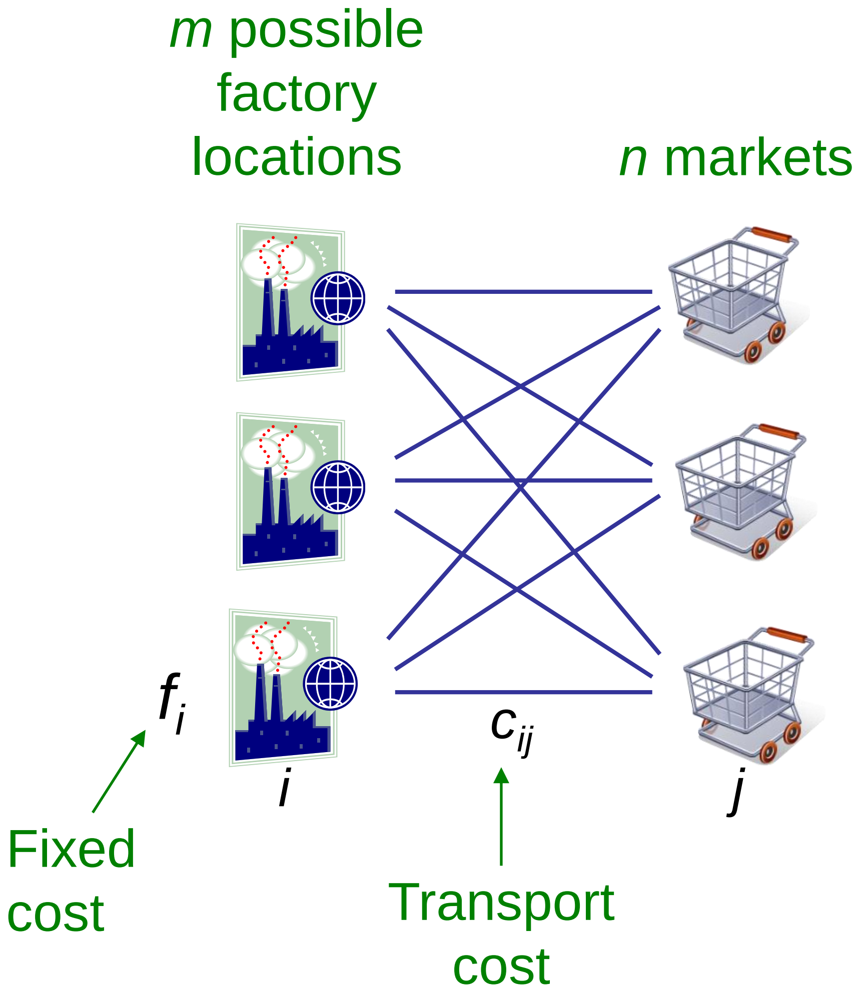
Recall that a disjunction of linear systems (representing polyhedra with the same recession cone)

$$\min \ cx$$
$$\bigvee_k \left(A^k x \geq b^k \right)$$

…has the MILP model

$$\min \ cx$$
$$A^k x^k \geq b^k y_k, \text{ all } k$$
$$\sum_k y_k = 1$$
$$x = \sum_k x^k$$
$$y_k \in \{0,1\}$$

Example: Uncapacitated facility location



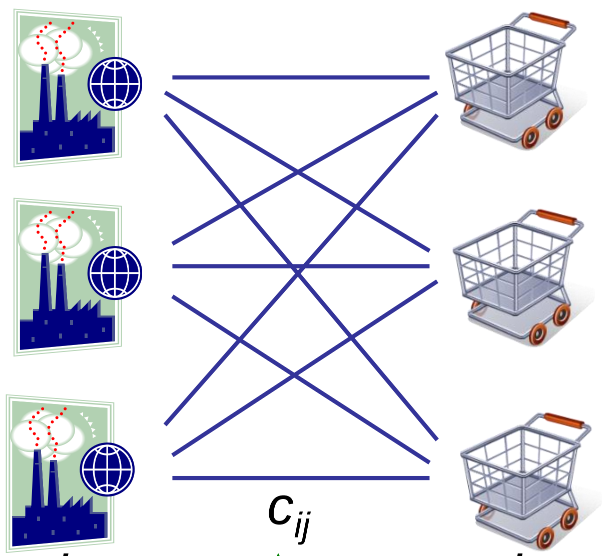
Locate factories to serve markets so as to minimize total fixed cost and transport cost.

No limit on production capacity of each factory.

Uncapacitated facility location

m possible factory locations

n markets



f_i i c_{ij} j

Fixed cost

Transport cost

Disjunctive model:

Fraction of market *j*'s demand satisfied from location *i*

$$\min \sum_i z_i + \sum_{ij} c_{ij} \boxed{x_{ij}}$$

$$\begin{pmatrix} x_{ij} = 0, \text{ all } j \\ z_i = 0 \end{pmatrix} \vee \begin{pmatrix} 0 \le x_{ij} \le 1, \text{ all } j \\ z_i \ge f_i \end{pmatrix}, \quad \text{all } i$$

$$\sum_i x_{ij} = 1, \text{ all } j$$

No factory at location *i*

Factory at location *i*

Uncapacitated facility location

MILP formulation:

$$\min \sum_i f_i y_i + \sum_{ij} c_{ij} x_{ij}$$

$$0 \le x_{ij} \le y_i, \quad \text{all } i, j$$

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

Based on LP relaxation of disjunction described earlier

Disjunctive model:

$$\min \sum_i z_i + \sum_{ij} c_{ij} x_{ij}$$

$$\begin{pmatrix} x_{ij} = 0, \ \text{all } j \\ z_i = 0 \end{pmatrix} \vee \begin{pmatrix} 0 \le x_{ij} \le 1, \ \text{all } j \\ z_i \ge f_i \end{pmatrix}, \quad \text{all } i$$

$$\sum_i x_{ij} = 1, \quad \text{all } j$$

No factory at location *i*

Factory at location *i*

Uncapacitated facility location

Maximum output from location i

MILP formulation:

$$\min \sum_i f_i y_i + \sum_{ij} c_{ij} x_{ij}$$
$$0 \le x_{ij} \le y_i, \quad \text{all } i, j$$
$$y_i \in \{0,1\}$$

Beginner's model:

$$\min \sum_i f_i y_i + \sum_{ij} c_{ij} x_{ij}$$
$$\sum_j x_{ij} \le \boxed{n y_i}, \quad \text{all } i, j$$
$$y_i \in \{0,1\}$$

Based on capacitated location model.

It has a **weaker continuous relaxation** (obtained by replacing $y_i \in \{0,1\}$ with $0 \le y_i \le 1$).

This beginner's mistake can be avoided by starting with disjunctive formulation.

Knapsack Modeling

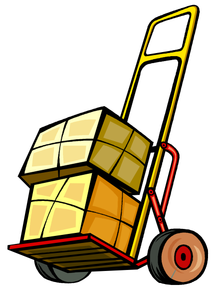
- Knapsack models consist of **knapsack covering** and **knapsack packing** constraints.
- The freight transfer model presented earlier is an example.
- We will consider a similar example that combines disjunctive and knapsack modeling.
- Most OR professionals are unlikely to write a model as good as the one presented here.

Note on tightness of knapsack models

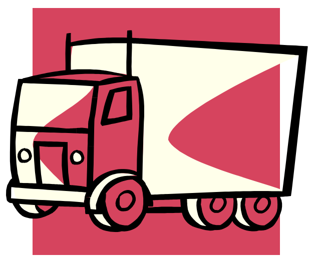
- The continuous relaxation of a knapsack model is not in general a convex hull relaxation.
  - A disjunctive formulation would provide a convex hull relaxation, but there are exponentially many disjuncts.
- Knapsack cuts can significantly tighten the relaxation.

Example: Package transport

Each package j has size a_j



Each truck i has capacity Q_i and costs c_i to operate



Disjunctive model

Knapsack constraints

$$\min \sum_i z_i$$

$$\sum_i Q_i y_i \geq \sum_j a_j; \quad \sum_i x_{ij} = 1, \text{ all } j$$

$$\begin{pmatrix} y_i = 1 \\ z_i = c_i \\ \sum_j a_j x_{ij} \leq Q_i \\ 0 \leq x_{ij} \leq 1, \text{ all } j \end{pmatrix} \vee \begin{pmatrix} y_i = 0 \\ z_i = 0 \\ x_{ij} = 0 \end{pmatrix}, \quad \text{all } i$$

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

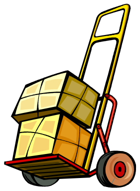
Truck i used

Truck i not used

1 if truck i carries package j

1 if truck i is used

Example: Package transport



MILP model

$$\min \sum_i c_i y_i$$

$$\sum_i Q_i y_i \geq \sum_j a_j; \quad \sum_i x_{ij} = 1, \text{ all } j$$

$$\sum_j a_j x_{ij} \leq Q_i y_i, \quad \text{all } i$$

$$x_{ij} \leq y_i, \quad \text{all } i, j$$

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

Disjunctive model

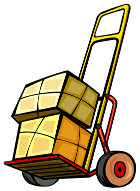
$$\min \sum_i z_i$$

$$\sum_i Q_i y_i \geq \sum_j a_j; \quad \sum_i x_{ij} = 1, \text{ all } j$$

$$\begin{pmatrix} y_i = 1 \\ z_i = c_i \\ \sum_j a_j x_{ij} \leq Q_i \\ 0 \leq x_{ij} \leq 1, \text{ all } j \end{pmatrix} \vee \begin{pmatrix} y_i = 0 \\ z_i = 0 \\ x_{ij} = 0 \end{pmatrix}, \quad \text{all } i$$

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

Example: Package transport



Most OR professionals would omit this constraint, since it is the sum over *i* of the next constraint. But it generates very effective knapsack cuts.

MILP model

$$\min \sum_i c_i y_i$$

$$\sum_i Q_i y_i \geq \sum_j a_j; \quad \sum_i x_{ij} = 1, \text{ all } j$$

$$\sum_j a_j x_{ij} \leq Q_i y_i, \text{ all } i$$

Modeling trick; unobvious without disjunctive approach

$$x_{ij} \leq y_i, \text{ all } i, j$$

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

Cutting Planes

0-1 Knapsack Cuts

Gomory Cuts

Mixed Integer Rounding Cuts

Example: Product Configuration

To review…

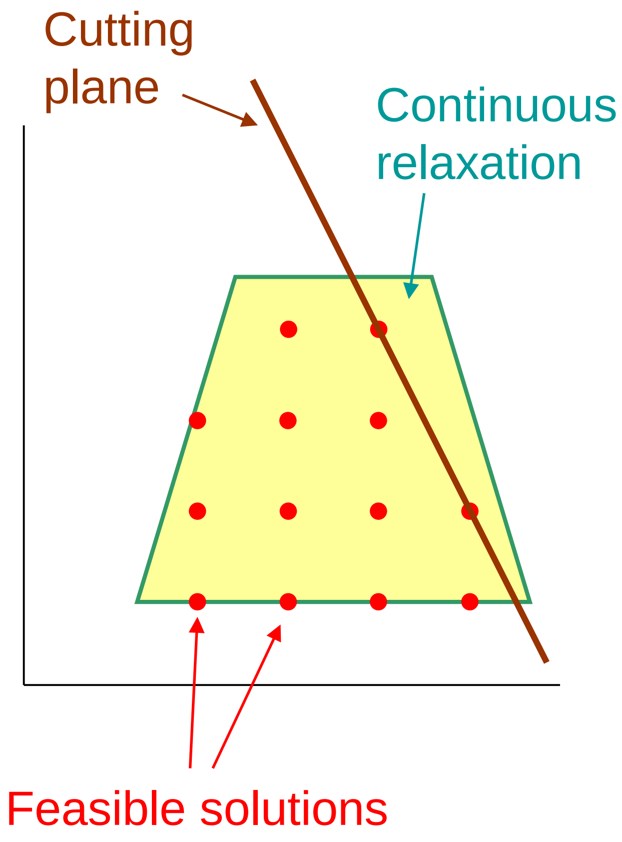
A **cutting plane** (cut, valid inequality) for an MILP model:

- …is **valid**

  - It is satisfied by all feasible solutions of the model.

- …**cuts off** solutions of the continuous relaxation.

  - This makes the relaxation tighter.



Motivation

- **Cutting planes** (cuts) tighten the continuous relaxation of an MILP model.
- **Knapsack cuts**
  - Generated for individual knapsack constraints.
  - We saw **general integer knapsack cuts** earlier.
  - **0-1 knapsack cuts** and **lifting** techniques are well studied and widely used.
- **Rounding cuts**
  - Generated for the entire MILP, they are widely used.
  - **Gomory cuts** for integer variables only.
  - **Mixed integer rounding cuts** for any MILP.

0-1 Knapsack Cuts

0-1 knapsack cuts are designed for knapsack constraints with 0-1 variables.

The analysis is different from that of general knapsack constraints, to exploit the special structure of 0-1 inequalities.

0-1 Knapsack Cuts

0-1 knapsack cuts are designed for knapsack constraints with 0-1 variables.

The analysis is different from that of general knapsack constraints, to exploit the special structure of 0-1 inequalities.

Consider a 0-1 knapsack packing constraint $ax \le a_0$. (Knapsack covering constraints are similarly analyzed.)

Index set J is a **cover** if $\sum_{j \in J} a_j > a_0$

The **cover inequality** $\sum_{j \in J} x_j \le |J| - 1$ is a **0-1 knapsack cut** for $ax \le a_0$

Only **minimal** covers need be considered.

Example

$J = \{1,2,3,4\}$ is a cover for

$$6x_1 + 5x_2 + 5x_3 + 5x_4 + 8x_5 + 3x_6 \le 17$$

This gives rise to the cover inequality

$$x_1 + x_2 + x_3 + x_4 \le 3$$

Index set J is a **cover** if $\sum_{j \in J} a_j > a_0$

The **cover inequality** $\sum_{j \in J} x_j \le |J| - 1$ is a **0-1 knapsack cut** for $ax \le a_0$

Only **minimal** covers need be considered.

Sequential lifting

- A cover inequality can often be strengthened by **lifting** it into a higher dimensional space.
    - That is, by adding variables.
- **Sequential lifting** adds one variable at a time.
- **Sequence-independent lifting** adds several variables at once.

Sequential lifting

To lift a cover inequality $\sum_{j \in J} x_j \leq |J| - 1$

add a term to the left-hand side $\sum_{j \in J} x_j + \pi_k x_k \leq |J| - 1$

where π_k is the largest coefficient for which the inequality is still valid.

So, $$\pi_k = |J| - 1 - \max_{\substack{x_j \in \{0,1\} \\ \text{for } j \in J}} \left\{ \sum_{j \in J} x_j \,\middle|\, \sum_{j \in J} a_j x_j \leq a_0 - a_k \right\}$$

This can be done repeatedly (by dynamic programming).

Example

Given $6x_1 + 5x_2 + 5x_3 + 5x_4 + 8x_5 + 3x_6 \le 17$

To lift $x_1 + x_2 + x_3 + x_4 \le 3$

add a term to the left-hand side $x_1 + x_2 + x_3 + x_4 + \pi_5 x_5 \le 3$

where

$$\pi_5 = 3 - \max_{\substack{x_j \in \{0,1\} \\ \text{for } j \in \{1,2,3,4\}}} \left\{ x_1 + x_2 + x_3 + x_4 \,\middle|\, 6x_1 + 5x_2 + 5x_3 + 5x_4 \le 17 - 8 \right\}$$

This yields $x_1 + x_2 + x_3 + x_4 + 2x_5 \le 3$

Further lifting leaves the cut unchanged.

But if the variables are added in the order x_6, x_5, the result is different:

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \le 3$$

Sequence-independent lifting

- Sequence-independent lifting usually yields a weaker cut than sequential lifting.
  - But it adds all the variables at once and is much faster.
  - Commonly used in commercial MILP solvers.

Sequence-independent lifting

To lift a cover inequality $\sum_{j \in J} x_j \le |J| - 1$

add terms to the left-hand side $\sum_{j \in J} x_j + \sum_{j \notin J} \rho(a_j) x_k \le |J| - 1$

where
$$\rho(u) = \begin{cases} j & \text{if } A_j \le u \le A_{j+1} - \Delta \text{ and } j \in \{0,\ldots,p-1\} \\ j + (u - A_j)/\Delta & \text{if } A_j - \Delta \le u < A_j - \Delta \text{ and } j \in \{1,\ldots,p-1\} \\ p + (u - A_p)/\Delta & \text{if } A_p - \Delta \le u \end{cases}$$

with
$$\Delta = \sum_{j \in J} a_j - a_0 \qquad A_j = \sum_{k=1}^{j} a_k$$
$$J = \{1,\ldots,p\} \qquad A_0 = 0$$

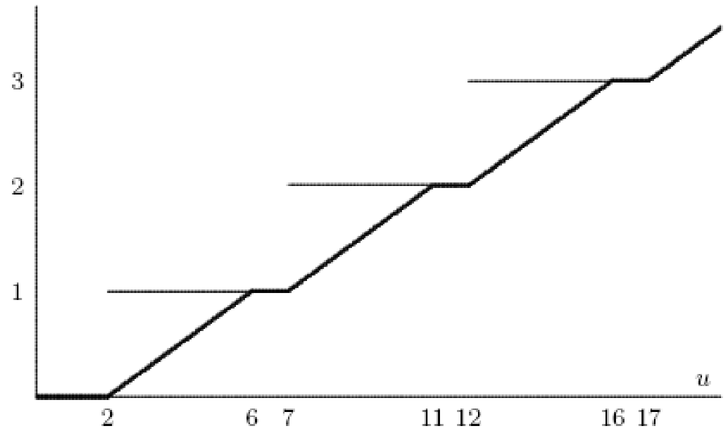
Example

Given $6x_1 + 5x_2 + 5x_3 + 5x_4 + 8x_5 + 3x_6 \leq 17$

To lift $x_1 + x_2 + x_3 + x_4 \leq 3$

Add terms $x_1 + x_2 + x_3 + x_4 + \rho(8)x_5 + \rho(3)x_6 \leq 3$

where $\rho(u)$ is given by



This yields the lifted cut

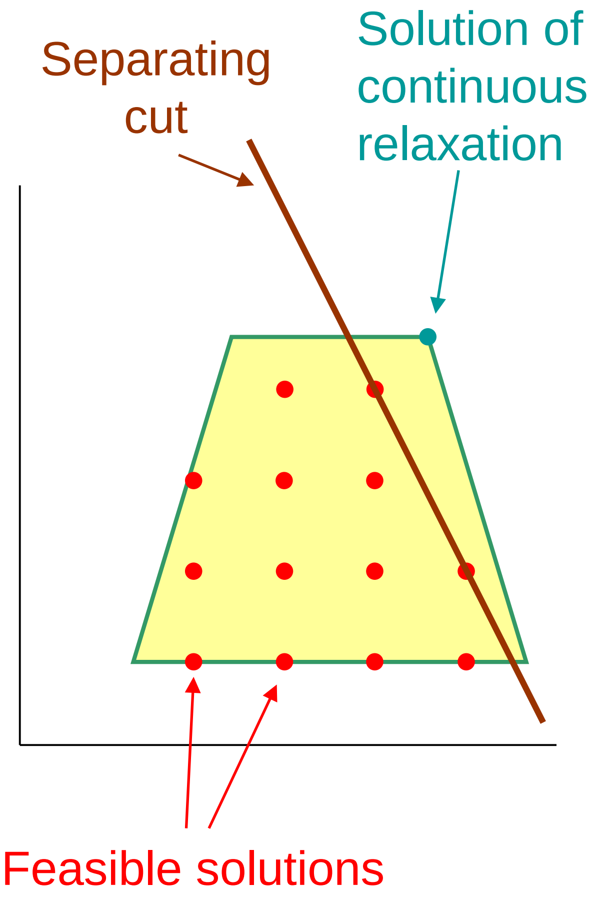
$$x_1 + x_2 + x_3 + x_4 + (5/4)x_5 + (1/4)x_6 \leq 3$$

Gomory Cuts

• When an integer programming problem has a nonintegral solution, we can generate at least one **Gomory cut** to cut off that solution.

- This is a special case of a **separating cut**, because it separates the current solution of the relaxation from the feasible set.

• Gomory cuts are widely used and very effective in MILP solvers.



Gomory cuts

Given an integer programming problem

$$\min cx$$
$$Ax = b$$
$$x \geq 0 \text{ and integral}$$

Let $(x_B, 0)$ be an optimal solution of the continuous relaxation, where

$$x_B = \hat{b} - \hat{N} x_N$$
$$\hat{b} = B^{-1} b, \quad \hat{N} = B^{-1} N$$

Then if x_i is nonintegral in this solution, the following **Gomory cut** is violated by $(x_B, 0)$:

$$x_i + \lfloor \hat{N}_i \rfloor x_N \leq \lfloor \hat{b}_i \rfloor$$

Example

$$\min\ 2x_1 + 3x_2$$
$$x_1 + 3x_2 \geq 3$$
$$4x_1 + 3x_2 \geq 6$$
$$x_1, x_2 \geq 0 \text{ and integral}$$

or

$$\min\ 2x_1 + 3x_2$$
$$x_1 + 3x_2 - x_3 = 3$$
$$4x_1 + 3x_2 - x_4 = 6$$
$$x_j \geq 0 \text{ and integral}$$

Optimal solution of the continuous relaxation has

$$x_B = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2/3 \end{bmatrix}$$

$$\hat{N} = \begin{bmatrix} 1/3 & -1/3 \\ -4/9 & 1/9 \end{bmatrix}$$

$$\hat{b} = \begin{bmatrix} 1 \\ 2/3 \end{bmatrix}$$

Example

$$\min\ 2x_1 + 3x_2$$
$$x_1 + 3x_2 \geq 3$$
$$4x_1 + 3x_2 \geq 6$$
$$x_1, x_2 \geq 0 \text{ and integral}$$

or

$$\min\ 2x_1 + 3x_2$$
$$x_1 + 3x_2 - x_3 = 3$$
$$4x_1 + 3x_2 - x_4 = 6$$
$$x_j \geq 0 \text{ and integral}$$

Optimal solution of the continuous relaxation has

$$x_B = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2/3 \end{bmatrix}$$

$$\hat{N} = \begin{bmatrix} 1/3 & -1/3 \\ -4/9 & 1/9 \end{bmatrix}$$

$$\hat{b} = \begin{bmatrix} 1 \\ 2/3 \end{bmatrix}$$

The Gomory cut $x_i + \lfloor \hat{N}_i \rfloor x_N \leq \lfloor \hat{b}_i \rfloor$

is $x_2 + \lfloor [-4/9 \quad 1/9] \rfloor \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \leq \lfloor 2/3 \rfloor$

or $x_2 - x_3 \leq 0$

In x_1, x_2 space this is $x_1 + 2x_2 \geq 3$

Example

$$\min\ 2x_1 + 3x_2$$
$$x_1 + 3x_2 \geq 3$$
$$4x_1 + 3x_2 \geq 6$$
$$x_1, x_2 \geq 0 \text{ and integral}$$

or

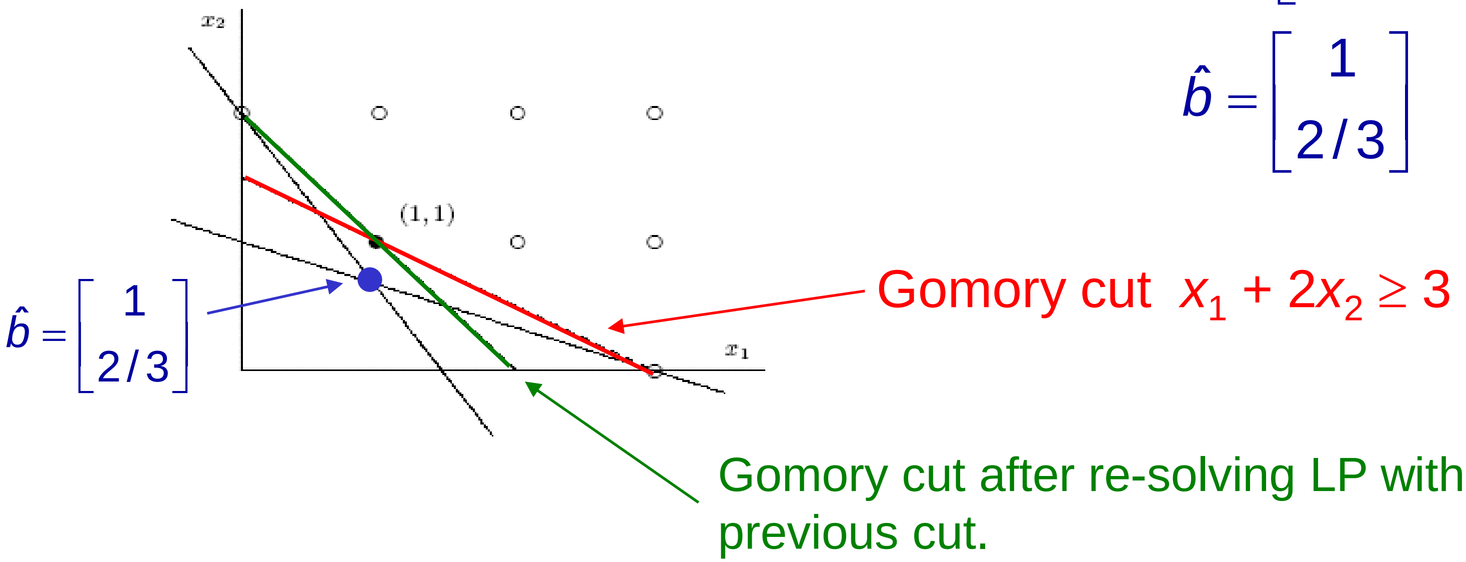
$$\min\ 2x_1 + 3x_2$$
$$x_1 + 3x_2 - x_3 = 3$$
$$4x_1 + 3x_2 - x_4 = 6$$
$$x_j \geq 0 \text{ and integral}$$

Optimal solution of the continuous relaxation has

$$x_B = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2/3 \end{bmatrix}$$

$$\hat{N} = \begin{bmatrix} 1/3 & -1/3 \\ -4/9 & 1/9 \end{bmatrix}$$

$$\hat{b} = \begin{bmatrix} 1 \\ 2/3 \end{bmatrix}$$



Mixed Integer Rounding Cuts

- **Mixed integer rounding** (MIR) **cuts** can be generated for solutions of any relaxed MILP in which one or more integer variables has a fractional value.
  - Like Gomory cuts, they are separating cuts.
  - MIR cuts are widely used in commercial solvers.

MIR cuts

Given an MILP problem

$$\min cx + dy$$
$$Ax + Dy = b$$
$$x, y \geq 0 \text{ and } y \text{ integral}$$

In an optimal solution of the continuous relaxation, let

$J = \{ j \mid y_j \text{ is nonbasic}\}$

$K = \{ j \mid x_j \text{ is nonbasic}\}$

N = nonbasic cols of $[A\ D]$

Then if y_i is nonintegral in this solution, the following **MIR cut** is violated by the solution of the relaxation:

$$y_i + \sum_{j \in J_1} \lceil \hat{N}_{ij} \rceil y_j + \sum_{j \in J_2} \left(\lfloor \hat{N}_{ij} \rfloor + \frac{\text{frac}(\hat{N}_{ij})}{\text{frac}(\hat{b}_i)} \right) + \frac{1}{\text{frac}(\hat{b}_i)} \sum_{j \in K} \hat{N}_{ij}^{+} x_j \geq \hat{N}_{ij} \lceil \hat{b}_i \rceil$$

where $J_1 = \left\{ j \in J \,\middle|\, \text{frac}(\hat{N}_{ij}) \geq \text{frac}(\hat{b}_j) \right\}$ $\qquad J_2 = J \setminus J_1$

Example

$$3x_1 + 4x_2 - 6y_1 - 4y_2 = 1$$
$$x_1 + 2x_2 - y_1 - y_2 = 3$$
$$x_j, y_j \geq 0, \quad y_j \text{ integer}$$

Take basic solution $(x_1, y_1) = (8/3, 17/3)$.

Then $\hat{N} = \begin{bmatrix} 1/3 & 2/3 \\ -2/3 & 8/3 \end{bmatrix}$ $\hat{b} = \begin{bmatrix} 8/3 \\ 17/3 \end{bmatrix}$

$J = \{2\}$, $K = \{2\}$, $J_1 = \emptyset$, $J_2 = \{2\}$

The MIR cut is $y_1 + \left(\lfloor 1/3 \rfloor + \frac{1/3}{2/3} \right) y_2 + \frac{1}{2/3} (2/3)^+ x_2 \geq \lceil 8/3 \rceil$

or $y_1 + (1/2) y_2 + x_2 \geq 3$

Lagrangean Relaxation

Lagrangean Duality
Properties of the Lagrangean Dual
Example: Fast Linear Programming
Domain Filtering
Example: Continuous Global Optimization

Motivation

- **Lagrangean relaxation** can provide better bounds than LP relaxation.
- The **Lagrangean dual** generalizes LP duality.
- It provides **domain filtering** analogous to that based on LP duality.
  - This is a key technique in **continuous global optimization**.
- Lagrangean relaxation gets rid of troublesome constraints by **dualizing** them.
  - That is, moving them into the objective function.
  - The Lagrangean relaxation may **decouple**.

Lagrangean Duality

Consider an inequality-constrained problem

$$\min \; f(x)$$

$$g(x) \geq 0$$ ← Hard constraints

$$x \in S$$ ← Easy constraints

The object is to get rid of (**dualize**) the hard constraints by moving them into the objective function.

Lagrangean Duality

Consider an inequality-constrained problem

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

It is related to an inference problem

$$\max\ v$$
$$g(x) \geq b \overset{s \in S}{\Longrightarrow} f(x) \geq v$$

implies

Lagrangean Dual problem: Find the tightest lower bound on the objective function that is implied by the constraints.

Primal

$$\min\ f(x)$$

$$g(x) \geq 0$$

$$x \in S$$

Dual

$$\max\ v$$

$$g(x) \geq b \overset{s \in S}{\Longrightarrow} f(x) \geq v$$

Surrogate

Let us say that

$g(x) \geq 0 \overset{x \in S}{\Longrightarrow} f(x) \geq v$ iff $\lambda g(x) \geq 0$ dominates $f(x) - v \geq 0$

for some $\lambda \geq 0$

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

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

Primal

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

Dual

$$\max \; v$$
$$g(x) \geq b \overset{s \in S}{\Rightarrow} f(x) \geq v$$

Surrogate

Let us say that

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

for some $\lambda \geq 0$

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

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

If we replace domination with material implication,
we get the surrogate dual, which gives better bounds
but lacks the nice properties of the Lagrangean dual.

Primal

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

Dual

$$\max \ v$$
$$g(x) \geq b \overset{s \in S}{\Rightarrow} f(x) \geq v$$

Let us say that

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

Surrogate

$\lambda g(x) \geq 0$ dominates $f(x) - v \geq 0$

for some $\lambda \geq 0$

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

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

Or $\quad v \leq \min_{x \in S} \{ f(x) - \lambda g(x) \}$

Primal

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

Dual

$$\max \ v$$
$$g(x) \geq b \overset{s \in S}{\Rightarrow} f(x) \geq v$$

Surrogate

Let us say that

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

for some $\lambda \geq 0$

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

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

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

So the dual becomes

$$\max \ v$$
$$v \leq \min_{x \in S} \{f(x) - \lambda g(x)\} \text{ for some } \lambda \geq 0$$

Now we have…

Primal

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

These constraints are **dualized**

Dual

$$\max \ v$$
$$v \leq \min_{x \in S} \{f(x) - \lambda g(x)\} \text{ for some } \lambda \geq 0$$

or

$$\max_{\lambda \geq 0} \theta(\lambda)$$

where

$$\theta(\lambda) = \min_{x \in S} \{f(x) - \lambda g(x)\}$$

Lagrangean relaxation

Vector of **Lagrange multipliers**

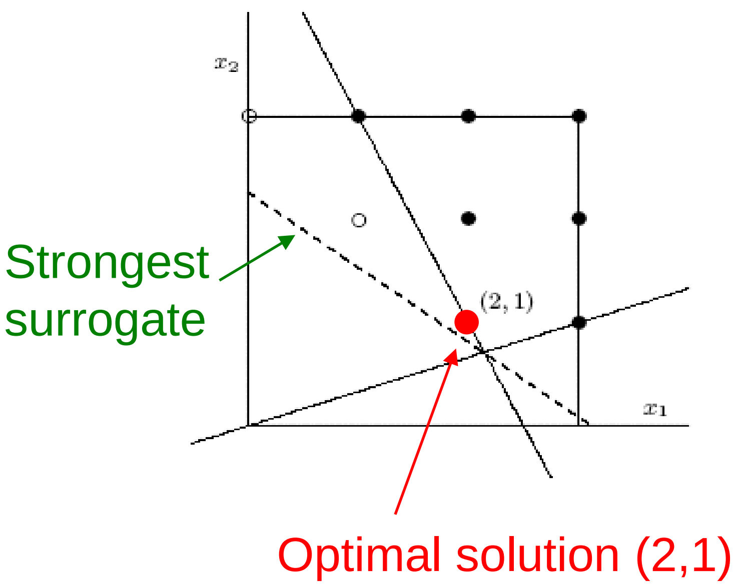
The Lagrangean dual can be viewed as the problem of finding the Lagrangean relaxation that gives the tightest bound.

Example

$$\min \ 3x_1 + 4x_2$$
$$-x_1 + 3x_2 \geq 0$$
$$2x_1 + x_2 - 5 \geq 0$$
$$x_1, x_2 \in \{0,1,2,3\}$$

The Lagrangean relaxation is

$$\theta(\lambda_1,\lambda_2) = \min_{x_j \in \{0,\ldots,3\}} \{3x_1 + 4x_2 - \lambda_1(-x_1 + 3x_2) - \lambda_2(2x_1 + x_2 - 5)\}$$
$$= \min_{x_j \in \{0,\ldots,3\}} \{(3 + \lambda_1 - 2\lambda_2)x_1 + (4 - 3\lambda_1 - \lambda_2)x_2 + 5\lambda_2\}$$



The Lagrangean relaxation is easy to solve for any given λ_1, λ_2:

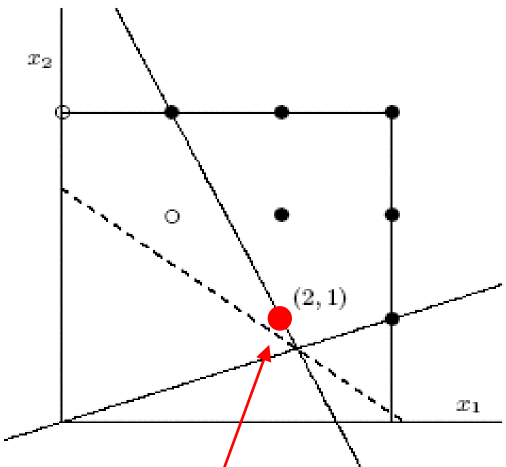
$$x_1 = \begin{cases} 0 & \text{if } 3 + \lambda_1 - 2\lambda_2 \geq 0 \\ 3 & \text{otherwise} \end{cases}$$

$$x_2 = \begin{cases} 0 & \text{if } 4 - 3\lambda_1 - \lambda_2 \geq 0 \\ 3 & \text{otherwise} \end{cases}$$

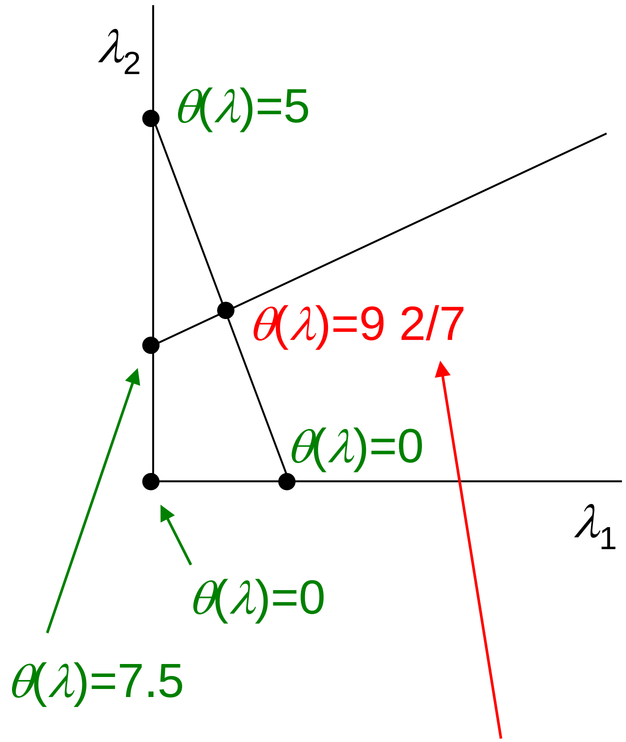
Example

$\theta(\lambda_1,\lambda_2)$ is piecewise linear and concave.

$$\min\ 3x_1 + 4x_2$$
$$-x_1 + 3x_2 \geq 0$$
$$2x_1 + x_2 - 5 \geq 0$$
$$x_1, x_2 \in \{0,1,2,3\}$$



Optimal solution (2,1)
Value = 10



Solution of Lagrangean dual:

(λ_1,λ_2) = (5/7, 13/7), $\theta(\lambda)$ = 9 2/7

Note **duality gap** between 10 and 9 2/7 (no strong duality).

Example

$$\min \; 3x_1 + 4x_2$$
$$-x_1 + 3x_2 \geq 0$$
$$2x_1 + x_2 - 5 \geq 0$$
$$x_1, x_2 \in \{0,1,2,3\}$$

Note: in this example, the Lagrangean dual provides the same bound (9 2/7) as the continuous relaxation of the IP.

This is because the Lagrangean relaxation can be solved as an LP:

$$\theta(\lambda_1,\lambda_2) = \min_{x_j \in \{0,\ldots,3\}} \left\{(3+\lambda_1-2\lambda_2)x_1 + (4-3\lambda_1-\lambda_2)x_2 + 5\lambda_2\right\}$$
$$= \min_{0 \leq x_j \leq 3} \left\{(3+\lambda_1-2\lambda_2)x_1 + (4-3\lambda_1-\lambda_2)x_2 + 5\lambda_2\right\}$$

Lagrangean duality is useful when the Lagrangean relaxation is tighter than an LP but nonetheless easy to solve.

Properties of the Lagrangean dual

Weak duality: For any feasible x^* and any $\lambda^* \geq 0$, $f(x^*) \geq \theta(\lambda^*)$.

In particular,

$$\begin{array}{l} \min \; f(x) \\ g(x) \geq 0 \\ x \in S \end{array} \quad \geq \quad \max_{\lambda \geq 0} \theta(\lambda)$$

Concavity: $\theta(\lambda)$ is concave. It can therefore be maximized by local search methods.

Complementary slackness: If x^* and λ^* are optimal, and there is no duality gap, then $\lambda^* g(x^*) = 0$.

Solving the Lagrangean dual

Let λ^k be the kth iterate, and let $\lambda^{k+1} = \lambda^k + \alpha_k \xi^k$

Subgradient of $\theta(\lambda)$ at $\lambda = \lambda^k$

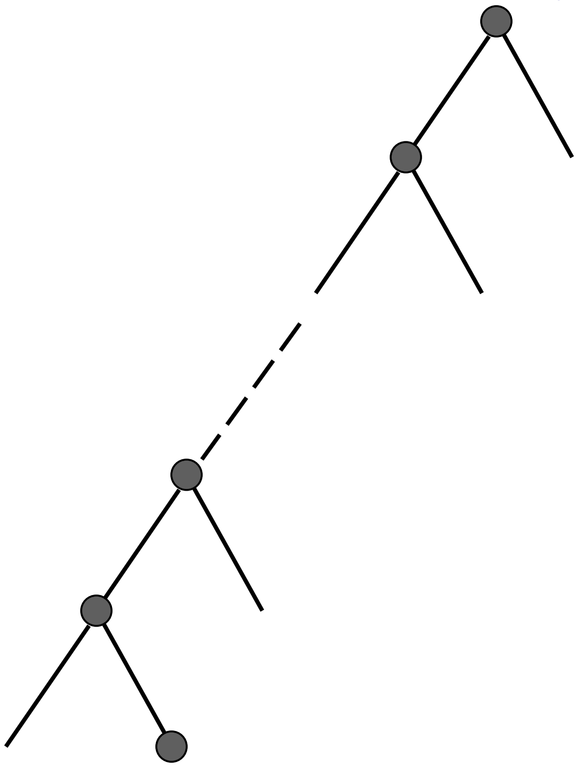
If x^k solves the Lagrangean relaxation for $\lambda = \lambda^k$, then $\xi^k = g(x^k)$.

This is because $\theta(\lambda) = f(x^k) + \lambda g(x^k)$ at $\lambda = \lambda^k$.

The stepsize α_k must be adjusted so that the sequence converges but not before reaching a maximum.

Example: Fast Linear Programming

- In CP contexts, it is best to process each node of the search tree very rapidly.
- Lagrangean relaxation may allow very fast calculation of a lower bound on the optimal value of the LP relaxation at each node.
- The idea is to solve the Lagrangean dual at the root node (which is an LP) and use the same Lagrange multipliers to get an LP bound at other nodes.



At root node, solve $\min \; cx$

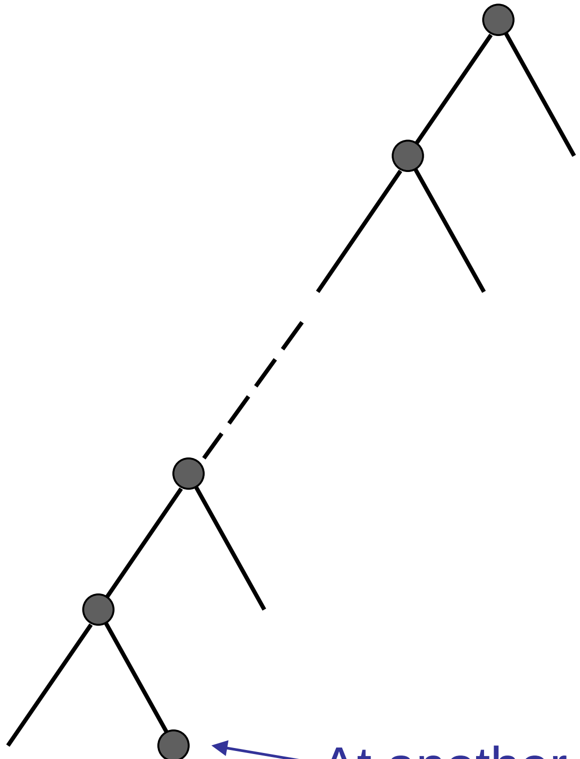
Dualize → $Ax \geq b \quad (\lambda)$

Special structure, e.g. variable bounds → $Dx \geq d$

$x \geq 0$

The (partial) LP dual solution λ^* solves the Lagrangean dual in which

$$\theta(\lambda) = \min_{\substack{Dx \geq d \\ x \geq 0}} \{ cx - \lambda(Ax - b) \}$$



At root node, solve

$$\begin{aligned} &\min\ cx \\ &Ax \geq b \quad (\lambda) \\ &Dx \geq d \\ &x \geq 0 \end{aligned}$$

Dualize → $Ax \geq b$

Special structure, e.g. variable bounds → $Dx \geq d$

The (partial) LP dual solution λ^* solves the Lagrangean dual in which

$$\theta(\lambda) = \min_{\substack{Dx \geq d \\ x \geq 0}} \{ cx - \lambda(Ax - b) \}$$

At another node, the LP is

$$\begin{aligned} &\min\ cx \\ &Ax \geq b \quad (\lambda) \\ &Dx \geq d \\ &Hx \geq h \\ &x \geq 0 \end{aligned}$$

Branching constraints, etc. → $Hx \geq h$

Here $\theta(\lambda^*)$ is still a lower bound on the optimal value of the LP and can be quickly calculated by solving a specially structured LP.

Domain Filtering

Suppose:

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

has optimal solution x^*, optimal value v^*, and optimal Lagrangean dual solution λ^*.

…and $\lambda_i^* > 0$, which means the i-th constraint is tight (complementary slackness);

…and the problem is a relaxation of a CP problem;

…and we have a feasible solution of the CP problem with value U, so that U is an upper bound on the optimal value.

Supposing $\begin{array}{l} \min\ f(x) \\ g(x) \geq 0 \\ x \in S \end{array}$ has optimal solution x^*, optimal value v^*, and optimal Lagrangean dual solution λ^*:

If x were to change to a value other than x^*, the LHS of i-th constraint $g_i(x) \geq 0$ would change by some amount Δ_i.

Since the constraint is tight, this would increase the optimal value as much as changing the constraint to $g_i(x) - \Delta_i \geq 0$.

So it would increase the optimal value at least $\lambda_i^* \Delta_i$.

(It is easily shown that Lagrange multipliers are marginal costs. Dual multipliers for LP are a special case of Lagrange multipliers.)

Supposing $\begin{array}{l} \min\ f(x) \\ g(x) \geq 0 \\ x \in S \end{array}$ has optimal solution x^*, optimal value v^*, and optimal Lagrangean dual solution λ^*:

We have found: a change in x that changes $g_i(x)$ by Δ_i increases the optimal value at least $\lambda_i^* \Delta_i$.

Since optimal value of this problem $\leq$ optimal value of the CP $\leq U$, we have $\lambda_i^* \Delta_i \leq U - v^*$, or

$$\Delta_i \leq \frac{U - v^*}{\lambda_i^*}$$

Supposing $\begin{array}{l} \min\ f(x) \\ g(x) \geq 0 \\ x \in S \end{array}$ has optimal solution x^*, optimal value v^*, and optimal Lagrangean dual solution λ^*:

We have found: a change in x that changes $g_i(x)$ by Δ_i increases the optimal value at least $\lambda_i^* \Delta_i$.

Since optimal value of this problem $\leq$ optimal value of the CP $\leq U$, we have $\lambda_i^* \Delta_i \leq U - v^*$, or

$$\Delta_i \leq \frac{U - v^*}{\lambda_i^*}$$

Since $\Delta_i = g_i(x) - g_i(x^*) = g_i(x)$, this implies the inequality

$$g_i(x) \leq \frac{U - v^*}{\lambda_i^*}$$

…which can be propagated.

Example: Continuous Global Optimization

- Some of the best continuous global solvers (e.g., BARON) combine OR-style relaxation with CP-style interval arithmetic and domain filtering.
- These methods can be combined with domain filtering based on Lagrange multipliers.

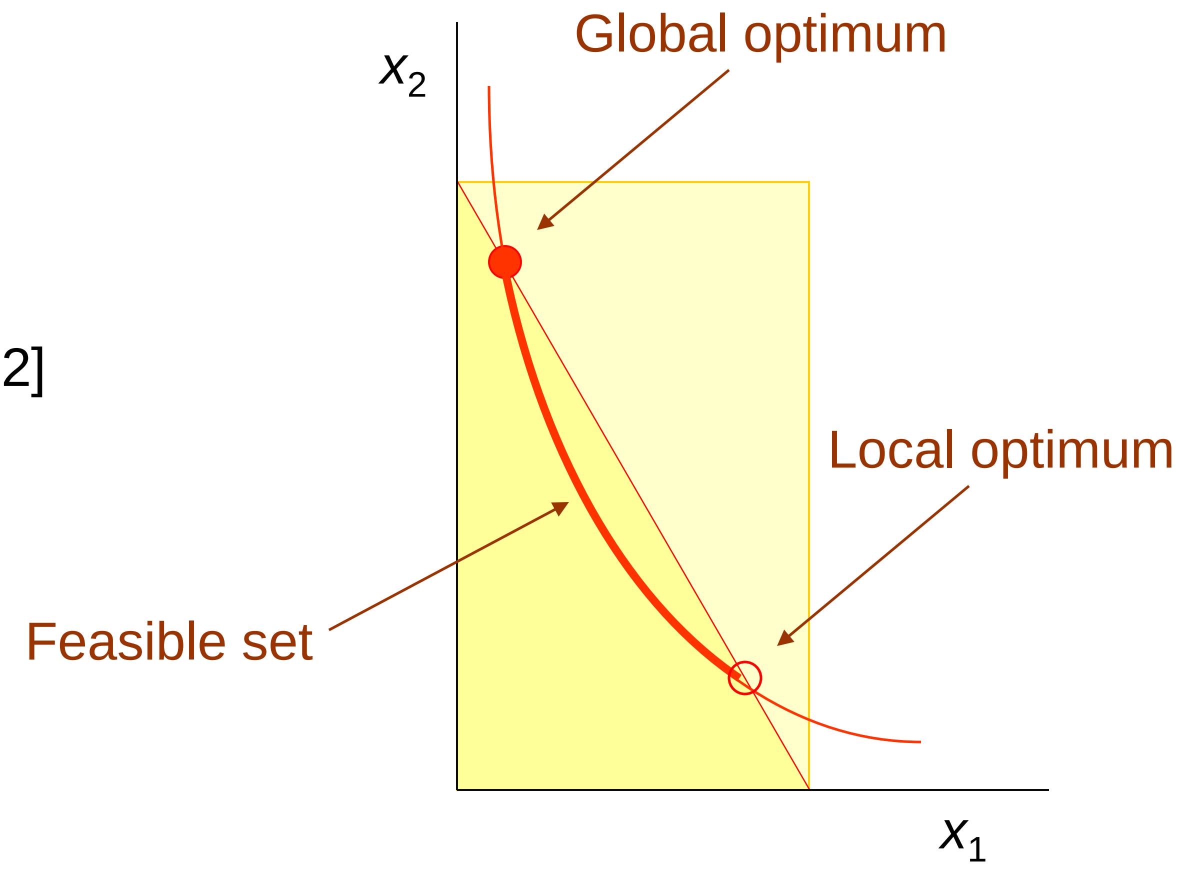
Continuous Global Optimization

$$\max \quad x_1 + x_2$$

$$4x_1x_2 = 1$$

$$2x_1 + x_2 \le 2$$

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

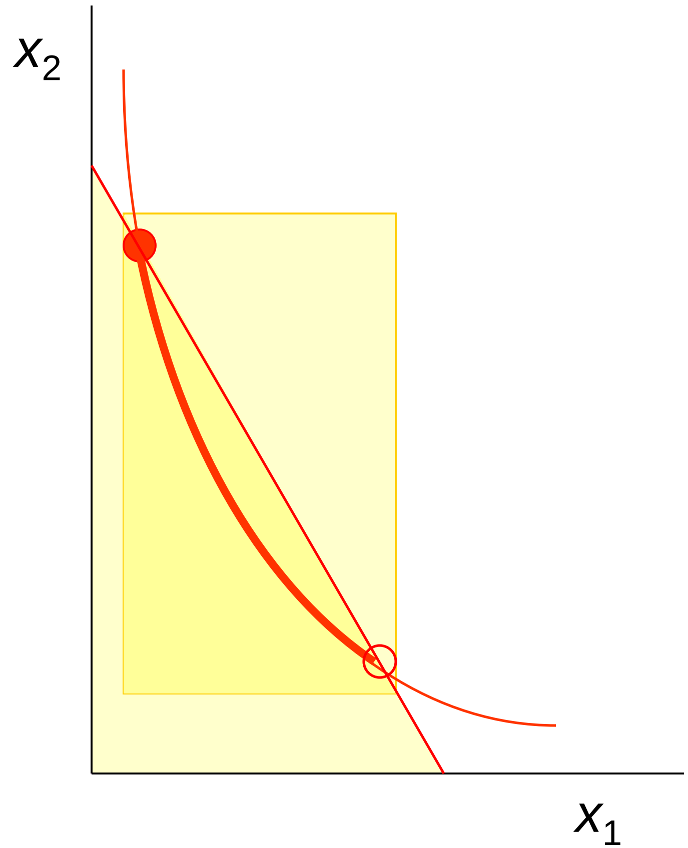


To solve it:

- **Search**: split interval domains of x_1, x_2.
  - Each **node** of search tree is a problem restriction.
- **Propagation:** Interval propagation, domain filtering.
  - Use **Lagrange multipliers** to infer valid inequality for propagation.
  - **Reduced-cost variable** fixing is a special case.
- **Relaxation:** Use **McCormick factorization** to obtain linear continuous relaxation.

Interval propagation

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



Relaxation (McCormick factorization)

Factor complex functions into elementary functions that have known linear relaxations.

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 (McCormick factorization)

Factor complex functions into elementary functions that have known linear relaxations.

For example, consider function $f(x) = x^2 \sin x$

Factor into elementary functions:

Let $y = x^2$, $z = \sin x$, $f(x) = yz$

Now write linear relaxations of the elementary functions.

Relaxation (McCormick factorization)

Factor complex functions into elementary functions that have known linear relaxations.

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:

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

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

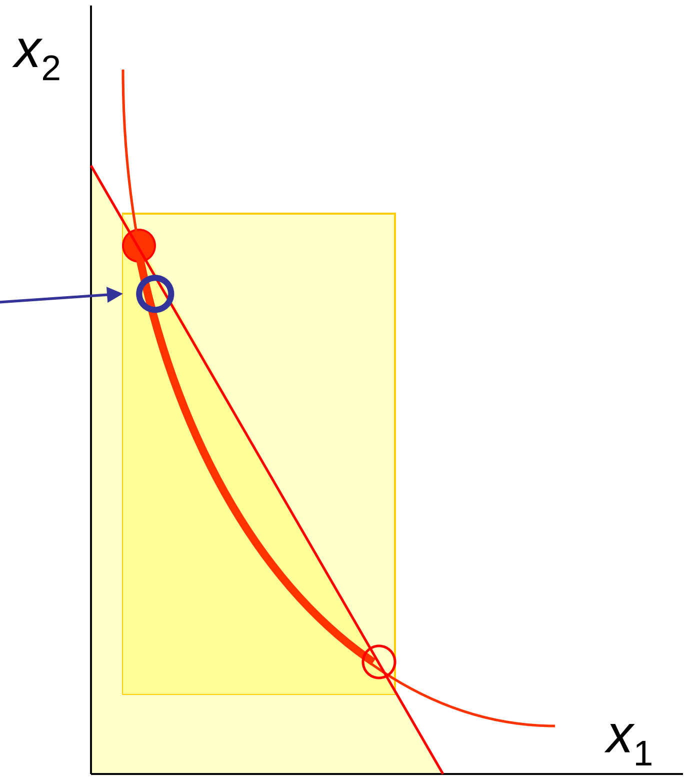
Relaxation (McCormick factorization)

The linear relaxation becomes:

$$
\begin{aligned}
&\min \;\; x_1 + x_2 \\
&4y = 1 \\
&2x_1 + x_2 \le 2 \\
&\underline{x}_2 x_1 + \underline{x}_1 x_2 - \underline{x}_1 \underline{x}_2 \le y \le \underline{x}_2 x_1 + \overline{x}_1 x_2 - \overline{x}_1 \underline{x}_2 \\
&\overline{x}_2 x_1 + \overline{x}_1 x_2 - \overline{x}_1 \overline{x}_2 \le y \le \overline{x}_2 x_1 + \underline{x}_1 x_2 - \underline{x}_1 \overline{x}_2 \\
&\underline{x}_j \le x_j \le \overline{x}_j, \;\; j = 1,2
\end{aligned}
$$

Relaxation (McCormick factorization)

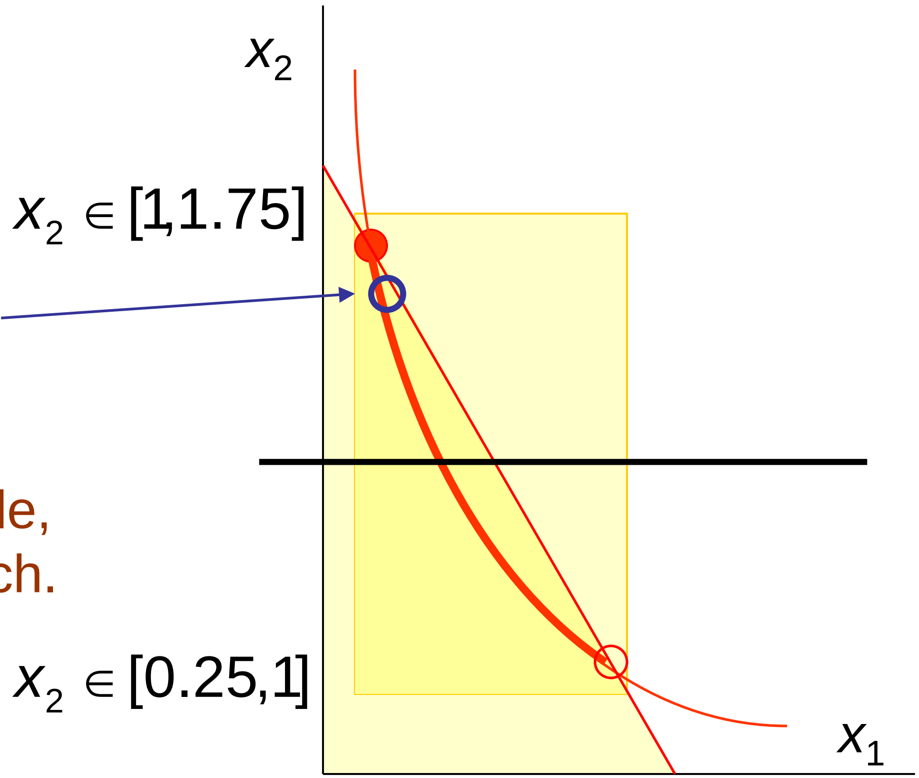
Solve linear relaxation.

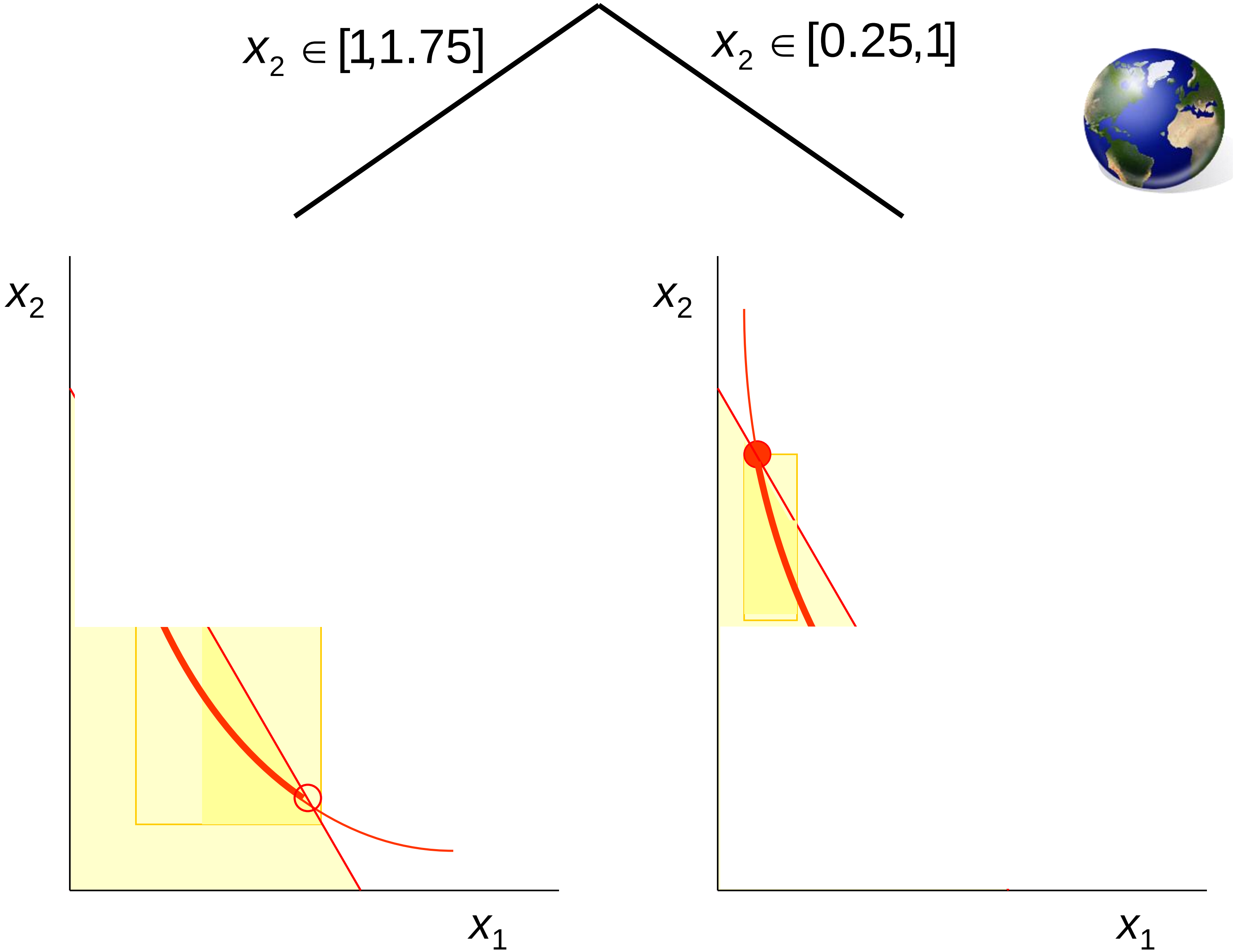


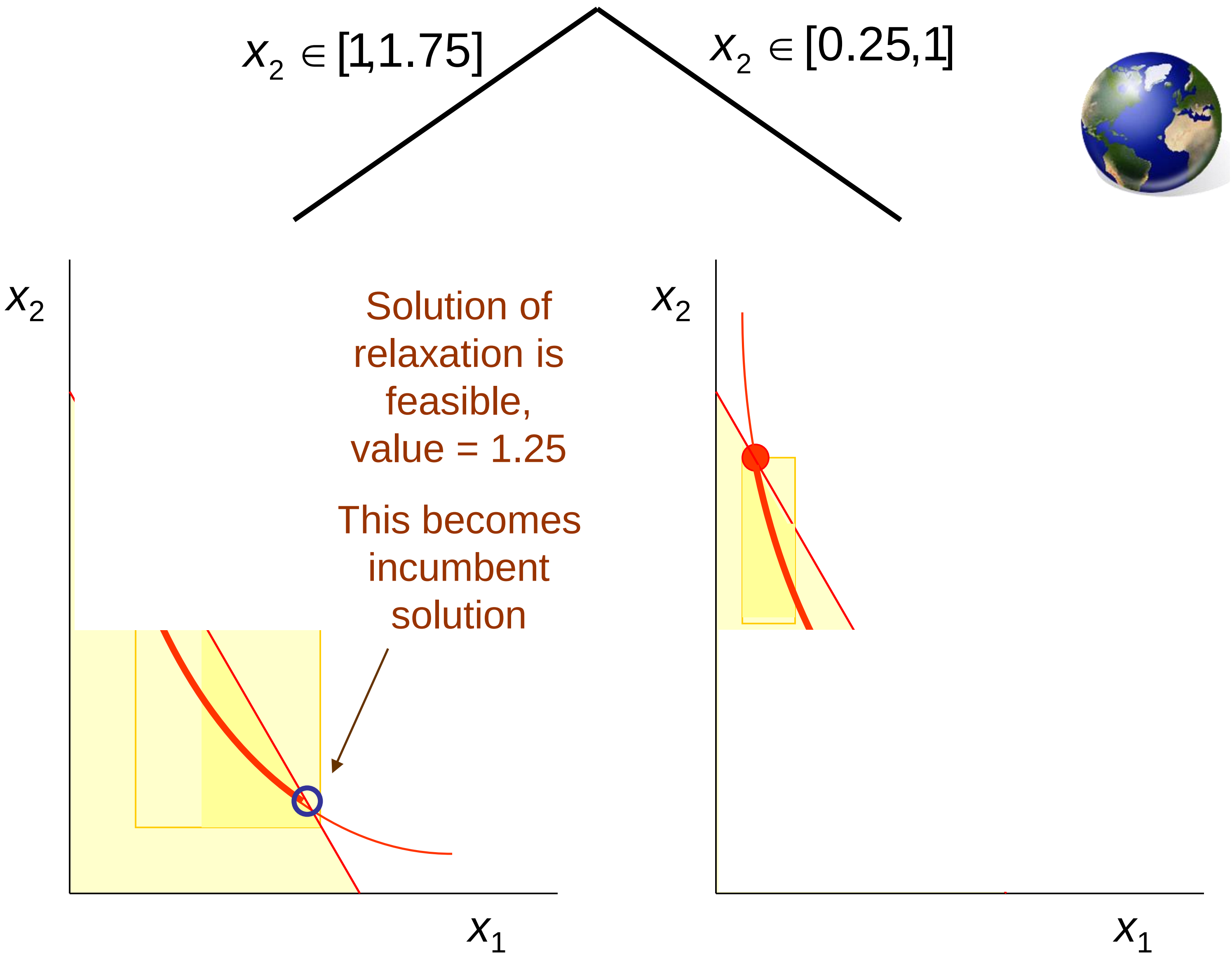
Relaxation (McCormick factorization)

Solve linear relaxation.

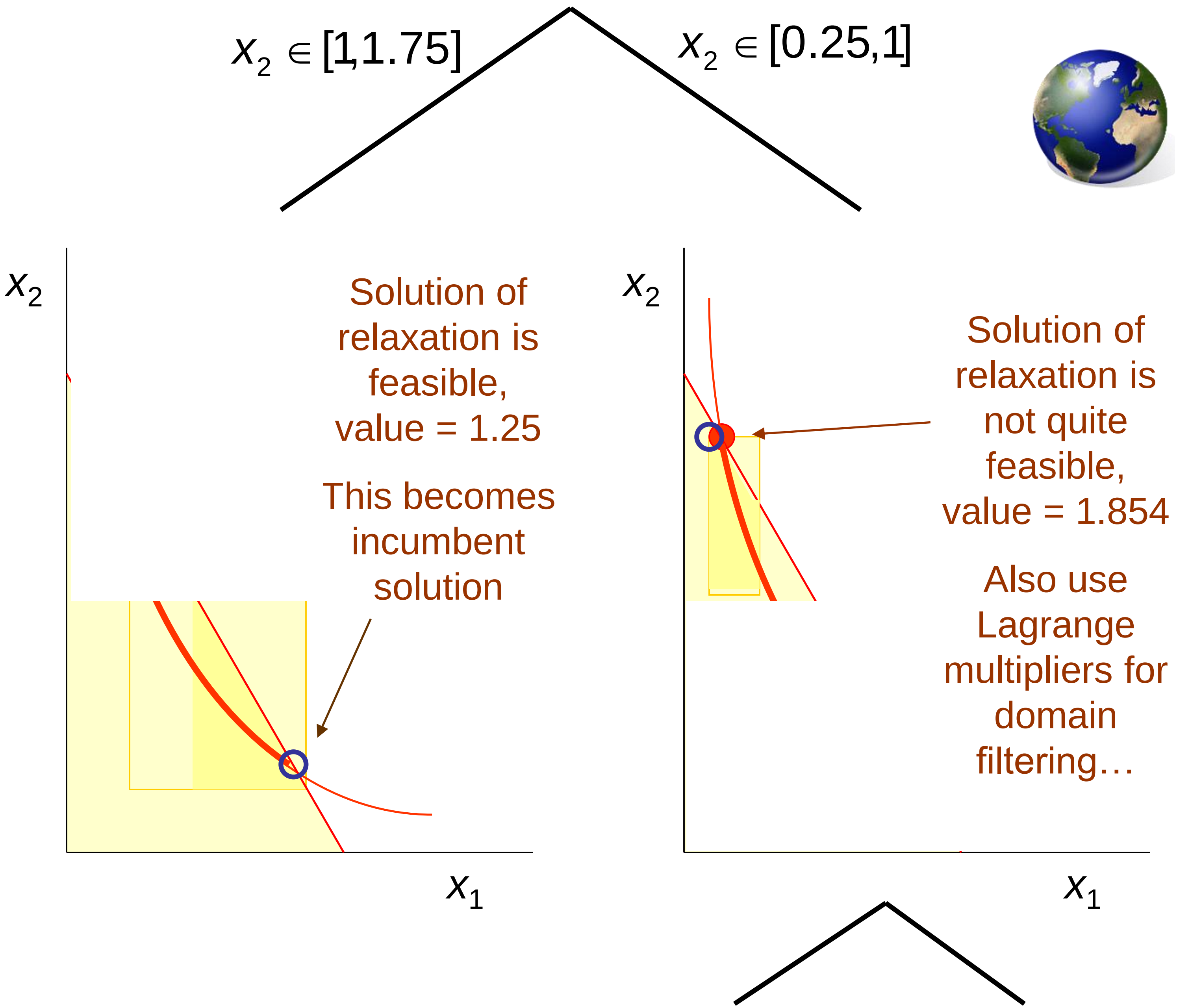
Since solution is infeasible,
split an interval and branch.




$x_2 \in [1,1.75]$
$x_2 \in [0.25,1]$
x_2
x_1
x_2
x_1



$x_2 \in [1, 1.75]$
$x_2 \in [0.25, 1]$
x_2
Solution of relaxation is feasible, value = 1.25
This becomes incumbent solution
x_1
x_2
x_1


$x_2 \in [1, 1.75]$
$x_2 \in [0.25, 1]$
x_2
x_1
Solution of relaxation is feasible, value = 1.25
This becomes incumbent solution
x_2
x_1
Solution of relaxation is not quite feasible, value = 1.854
Also use Lagrange multipliers for domain filtering…

Relaxation (McCormick factorization)

$$\min \; x_1 + x_2$$
$$4y = 1$$
$$2x_1 + x_2 \le 2$$
$$\underline{x}_2 x_1 + \underline{x}_1 x_2 - \underline{x}_1 \underline{x}_2 \le y \le \underline{x}_2 x_1 + \bar{x}_1 x_2 - \bar{x}_1 \underline{x}_2$$
$$\bar{x}_2 x_1 + \bar{x}_1 x_2 - \bar{x}_1 \bar{x}_2 \le y \le \bar{x}_2 x_1 + \underline{x}_1 x_2 - \underline{x}_1 \bar{x}_2$$
$$\underline{x}_j \le x_j \le \bar{x}_j, \quad j = 1,2$$

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

Relaxation (McCormick factorization)

$$\min \; x_1 + x_2$$
$$4y = 1$$
$$2x_1 + x_2 \le 2$$
$$\underline{x}_2 x_1 + \underline{x}_1 x_2 - \underline{x}_1 \underline{x}_2 \le y \le \underline{x}_2 x_1 + \bar{x}_1 x_2 - \bar{x}_1 \underline{x}_2$$
$$\bar{x}_2 x_1 + \bar{x}_1 x_2 - \bar{x}_1 \bar{x}_2 \le y \le \bar{x}_2 x_1 + \underline{x}_1 x_2 - \underline{x}_1 \bar{x}_2$$
$$\underline{x}_j \le x_j \le \bar{x}_j, \quad j = 1,2$$

Associated Lagrange multiplier in solution of relaxation is λ_2 = 1.1

This yields a valid inequality for propagation:

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

Value of relaxation

Lagrange multiplier

Value of incumbent solution

Constraint Programming Concepts

Domain Consistency

Cumulative Scheduling

Domain Consistency

- Also known as **generalized arc consistency**.
- A constraint set is **domain consistent** if every value in every variable domain is part of some feasible solution.
  - That is, the domains are reduced as much as possible.
  - Domain reduction is CP's biggest engine.

Domain Consistency

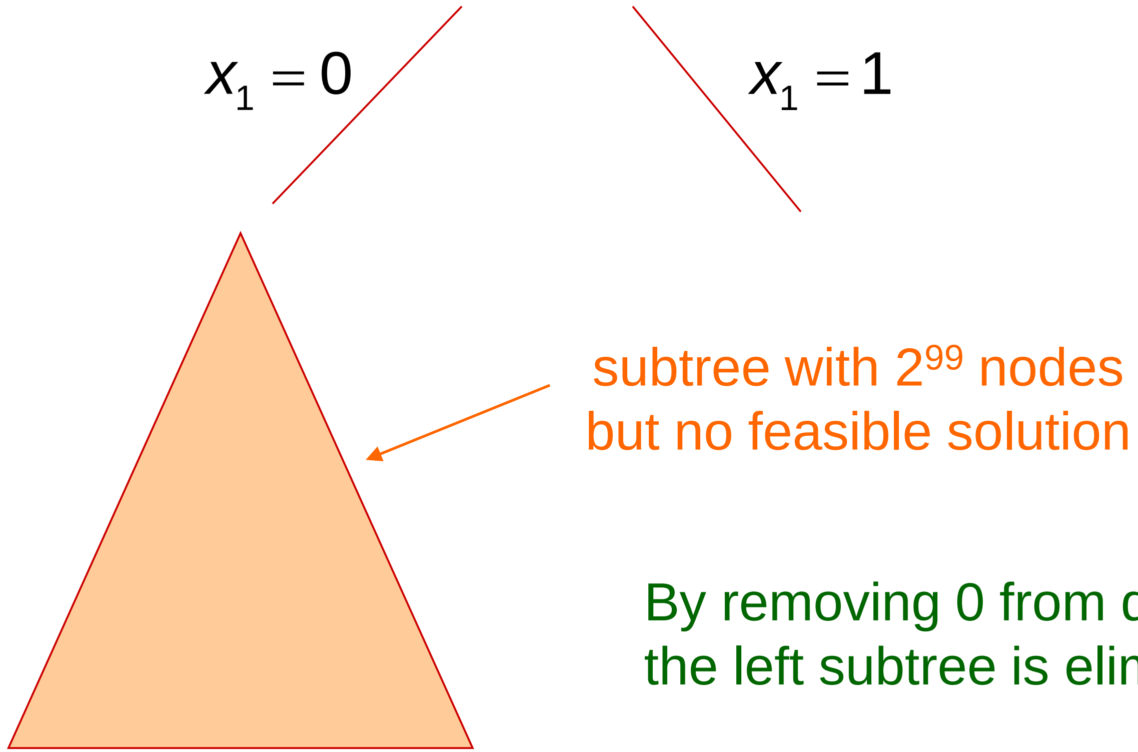
Consider the constraint set

$$x_1 + x_{100} \geq 1$$
$$x_1 - x_{100} \geq 0$$
$$x_j \in \{0,1\}$$

It is not domain consistent, because 0 appears in the domain of x_1, and yet no solution has $x_1 = 0$.

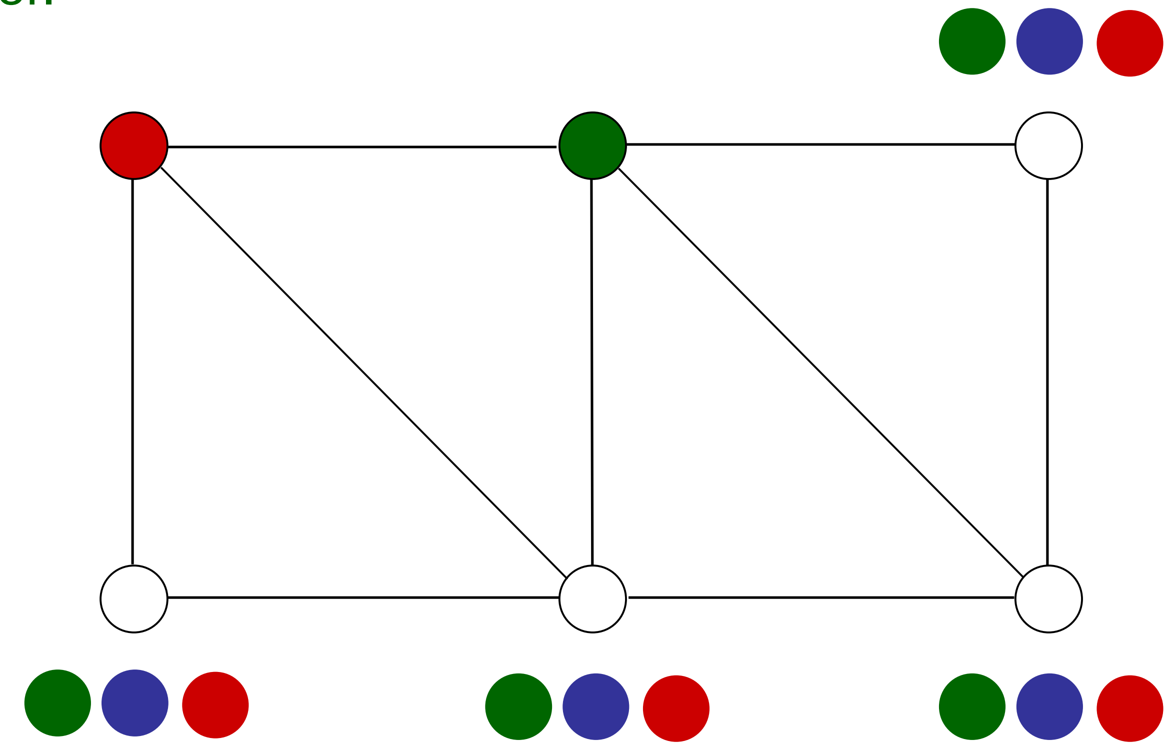
Removing 0 from domain of $x_1 = 1$ makes the set domain consistent.

$$x_1 + x_{100} \geq 1$$
$$x_1 - x_{100} \geq 1$$
other constraints
$$x_j \in \{0,1\}$$

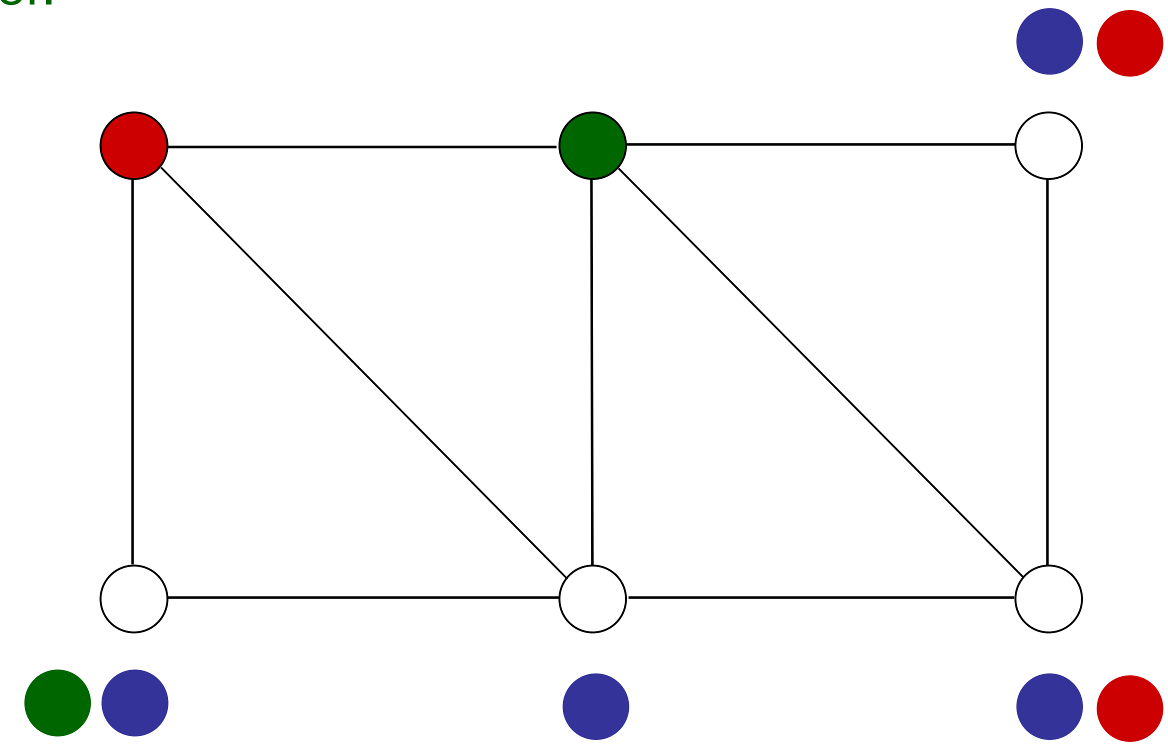


By removing 0 from domain of x_1,
the left subtree is eliminated

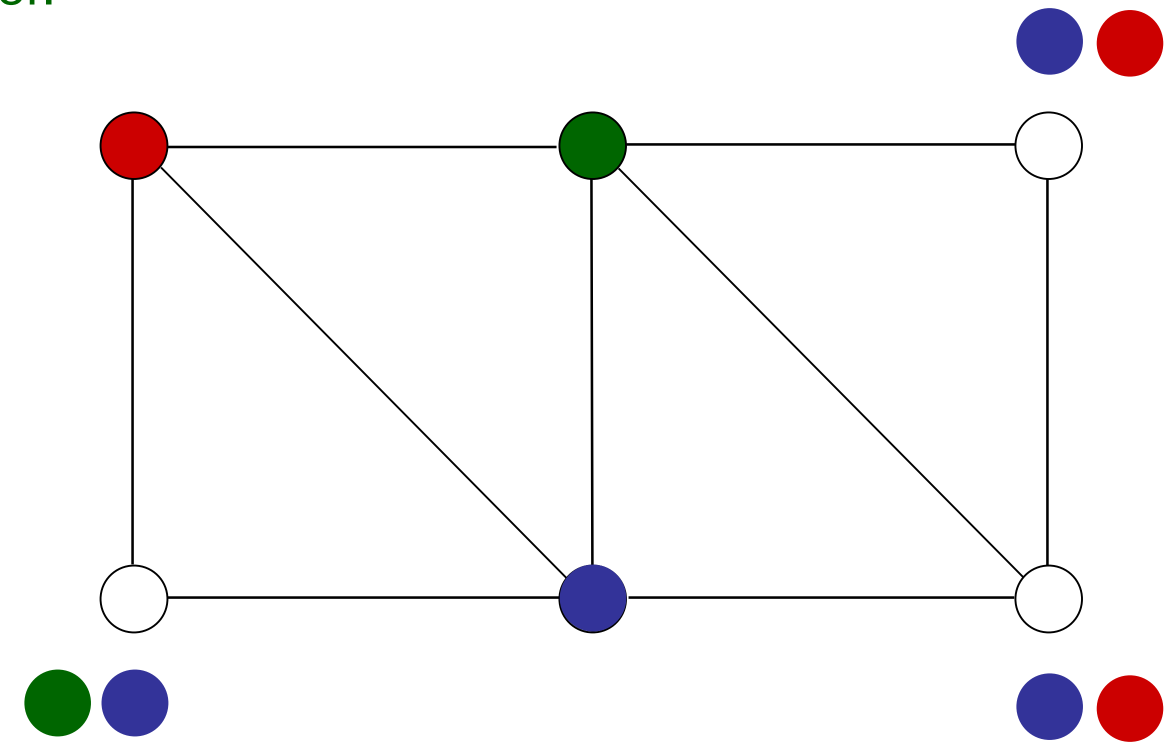
Graph coloring problem that can be solved by domain consistency maintenance alone. Color nodes with red, green, blue with no two adjacent nodes having the same color.



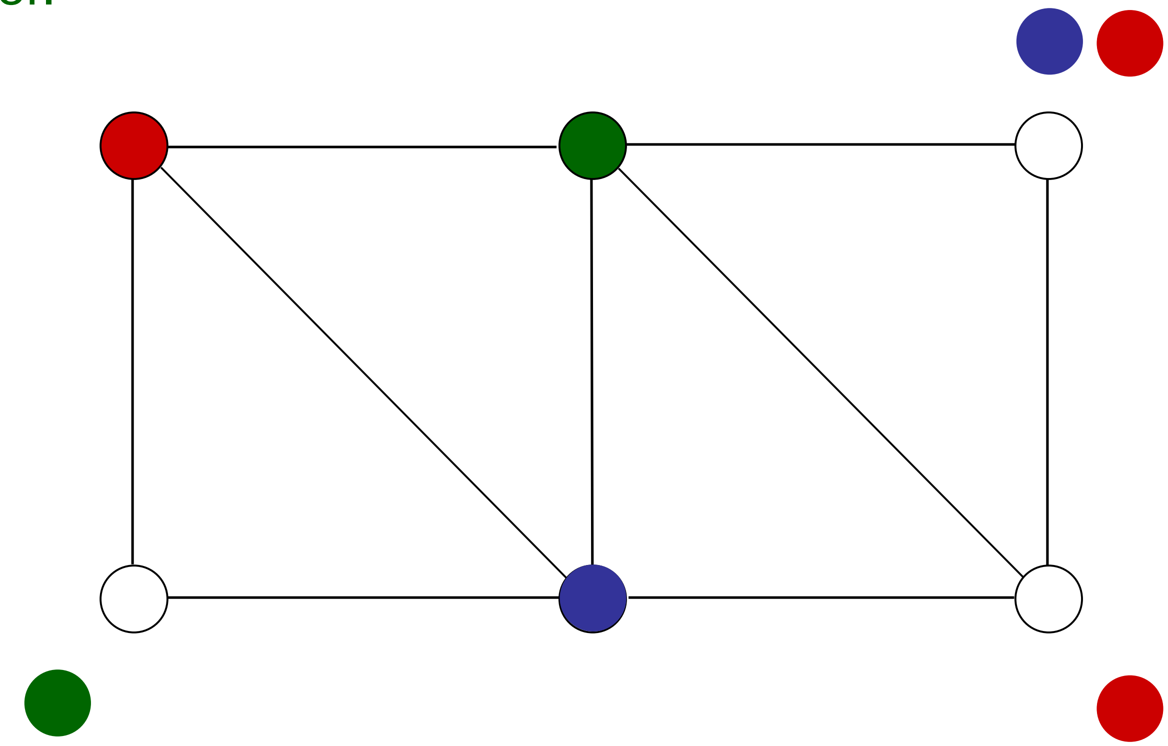
Graph coloring problem that can be solved by domain consistency maintenance alone. Color nodes with red, green, blue with no two adjacent nodes having the same color.



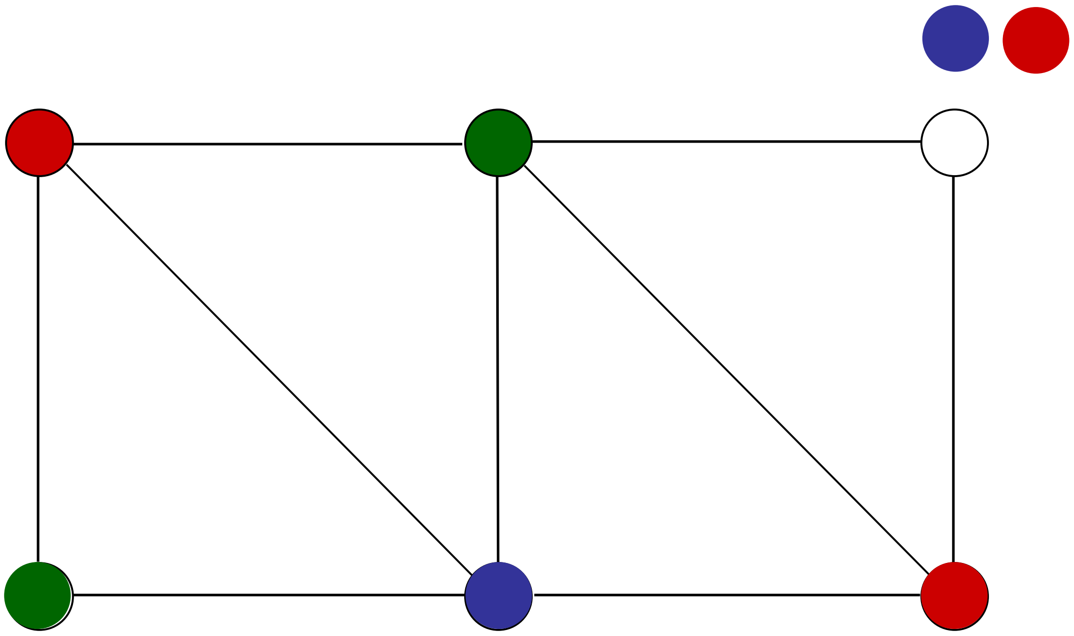
Graph coloring problem that can be solved by domain consistency maintenance alone. Color nodes with red, green, blue with no two adjacent nodes having the same color.



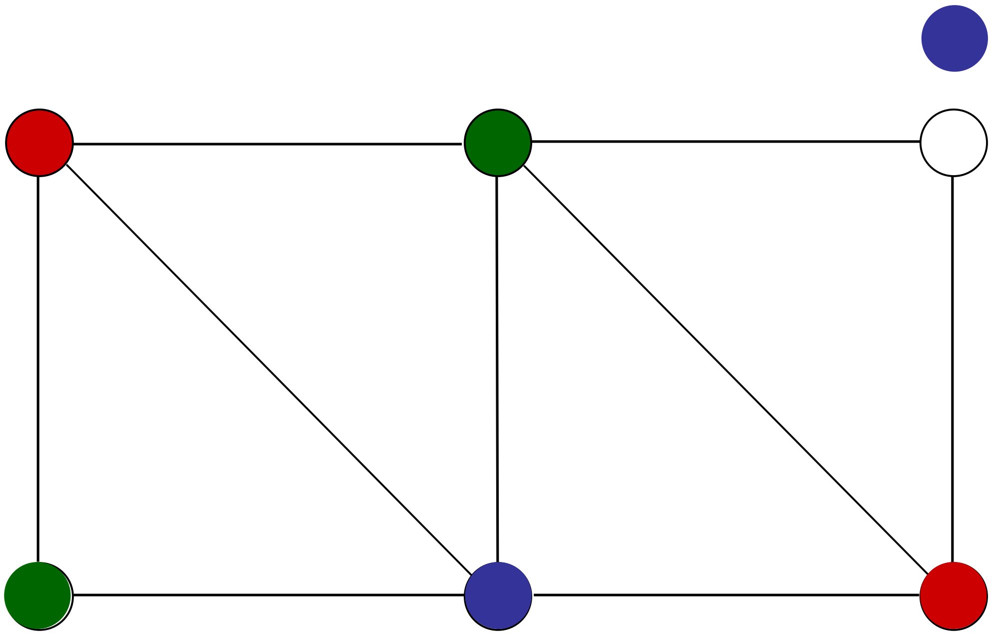
Graph coloring problem that can be solved by domain consistency maintenance alone. Color nodes with red, green, blue with no two adjacent nodes having the same color.



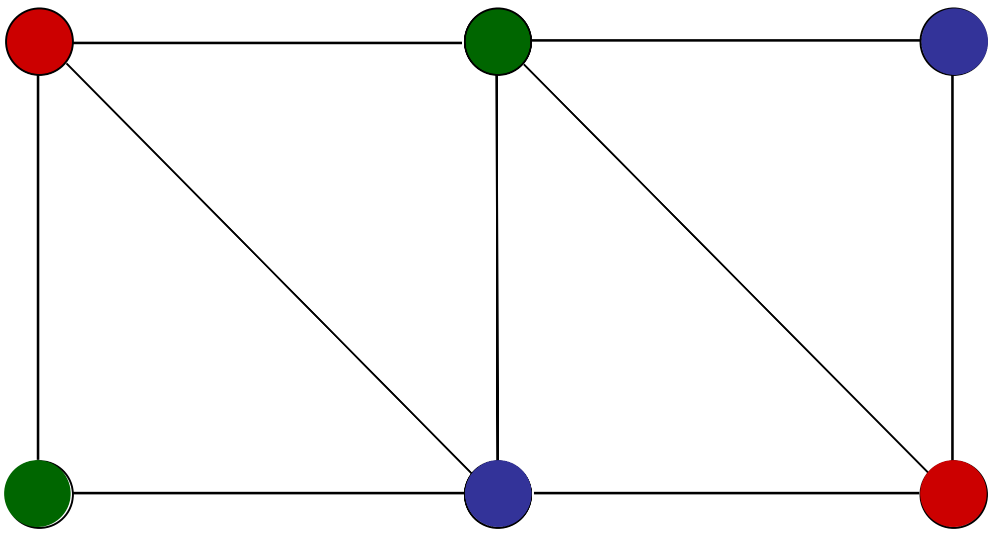
Graph coloring problem that can be solved by domain consistency maintenance alone. Color nodes with red, green, blue with no two adjacent nodes having the same color.



Graph coloring problem that can be solved by domaim consistency maintenance alone. Color nodes with red, green, blue with no two adjacent nodes having the same color.



Graph coloring problem that can be solved by domain consistency maintenance alone. Color nodes with red, green, blue with no two adjacent nodes having the same color.



Cumulative scheduling constraint

- Used for resource-constrained scheduling.
- Total resources consumed by jobs at any one time must not exceed *L*.

$$\text{cumulative}\big((t_1,\ldots,t_n),(p_1,\ldots,p_n),(c_1,\ldots,c_n),L\big)$$

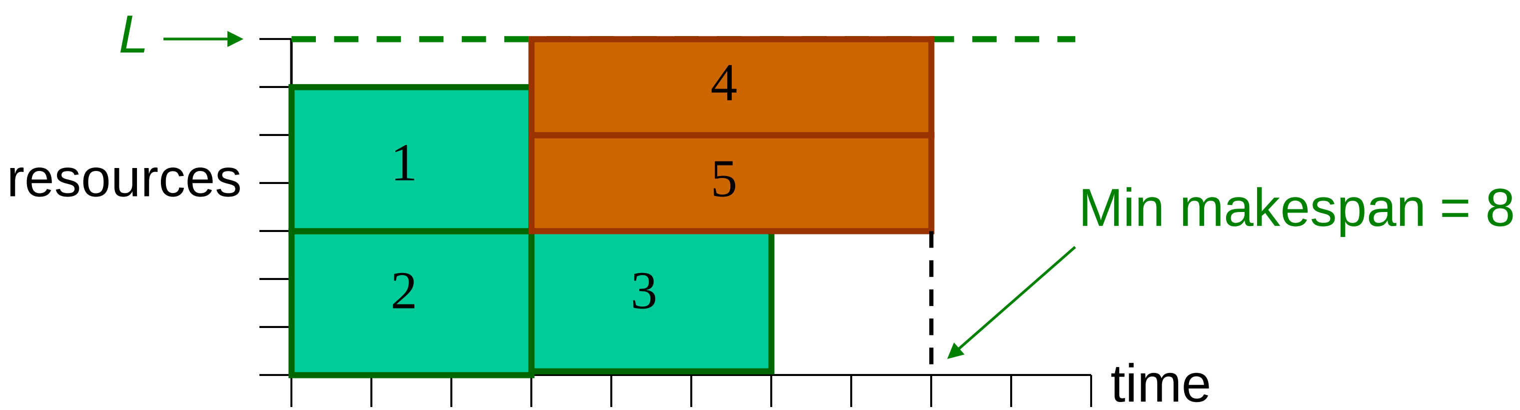
Job start times (variables)

Job processing times

Job resource requirements

Cumulative scheduling constraint

Minimize makespan (no deadlines, all release times = 0):



$$\begin{aligned}
\text{min} \quad & z \\
\text{s.t.} \quad & \text{cumulative}\big((t_1,\ldots,t_5),(3,3,3,5,5),(3,3,3,2,2),7\big) \\
& z \geq t_1 + 3 \\
& \vdots \\
& z \geq t_5 + 2
\end{aligned}$$

L → 7

Job start times → $(t_1,\ldots,t_5)$

Processing times → $(3,3,3,5,5)$

Resources used → $(3,3,3,2,2)$

CP Filtering Algorithms

All-different
Disjunctive Scheduling
Cumulative Scheduling

Filtering for all-different

$$\text{alldiff}\left(y_1,\ldots,y_n\right)$$

Domains can be filtered with an algorithm based on maximum cardinality bipartite matching and a theorem of Berge.

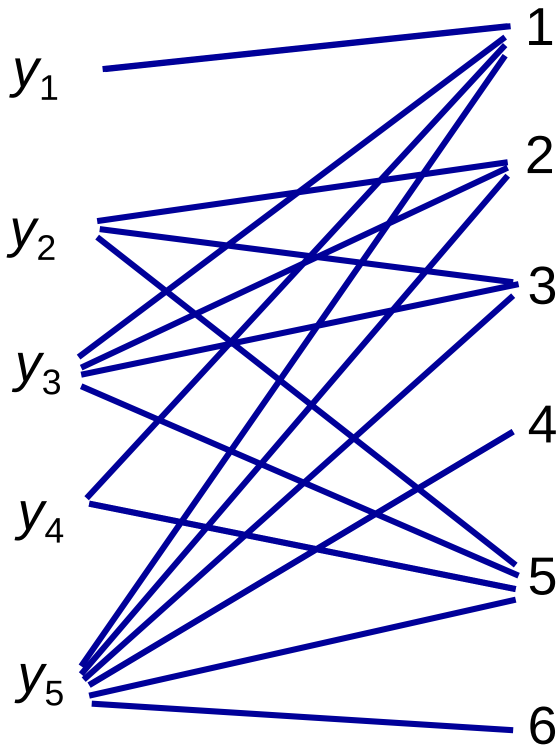
It is a special case of optimality conditions for max flow.

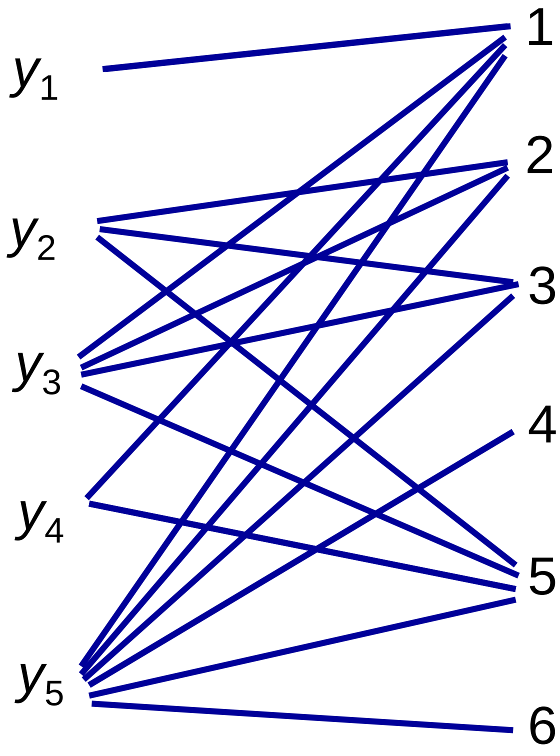
Filtering for alldiff

Consider the domains

$$y_1 \in \{1\}$$
$$y_2 \in \{2, 3, 5\}$$
$$y_3 \in \{1, 2, 3, 5\}$$
$$y_4 \in \{1, 5\}$$
$$y_5 \in \{1, 2, 3, 4, 5, 6\}$$

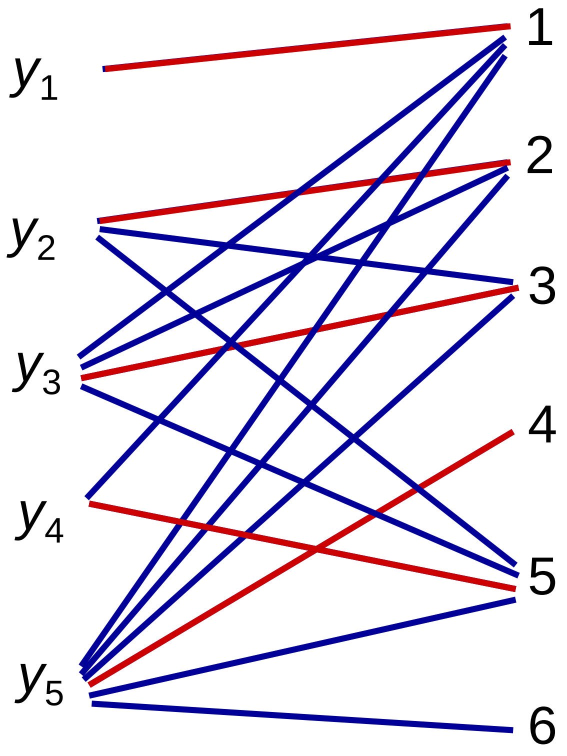
Indicate domains with edges





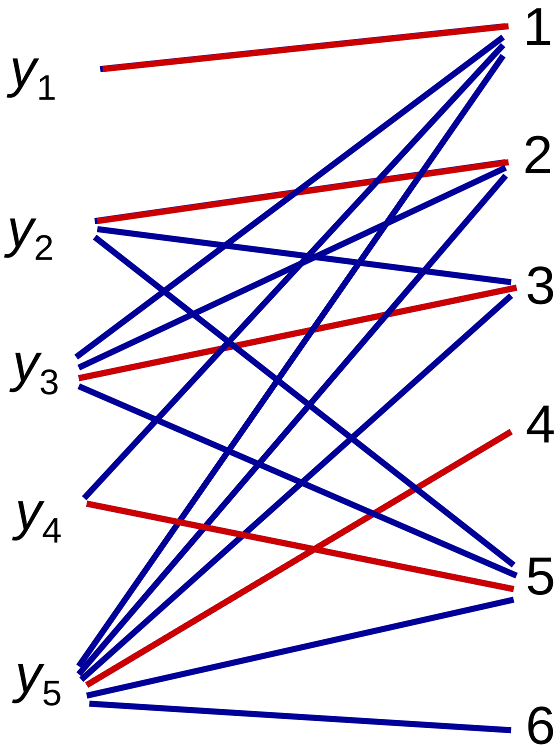
Indicate domains with edges

Find maximum cardinality bipartite matching.



Indicate domains with edges

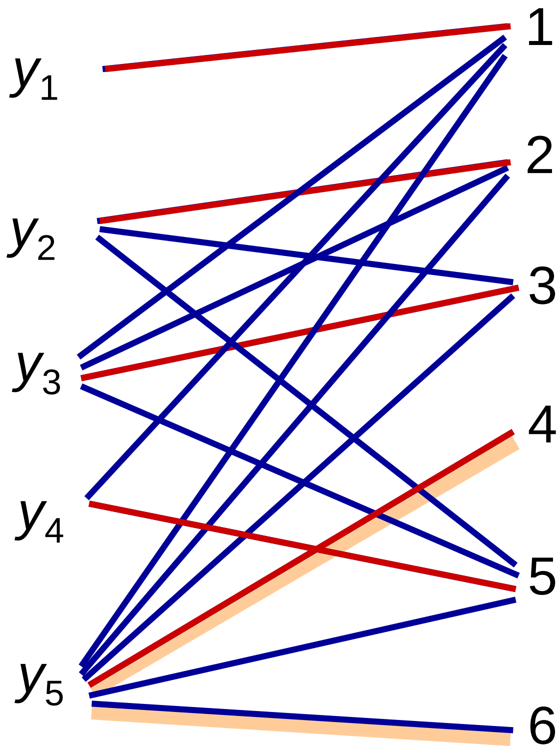
Find maximum cardinality bipartite matching.



Indicate domains with edges

Find maximum cardinality bipartite matching.

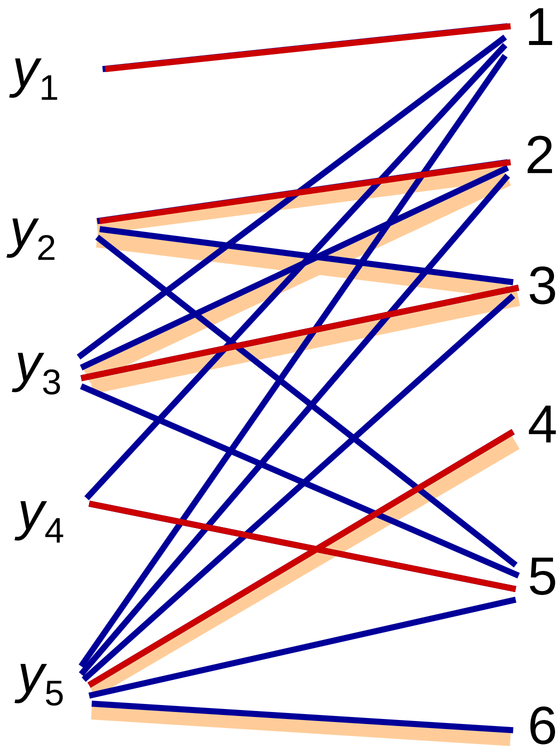
Mark edges in alternating paths that start at an uncovered vertex.



Indicate domains with edges

Find maximum cardinality bipartite matching.

Mark edges in alternating paths that start at an uncovered vertex.

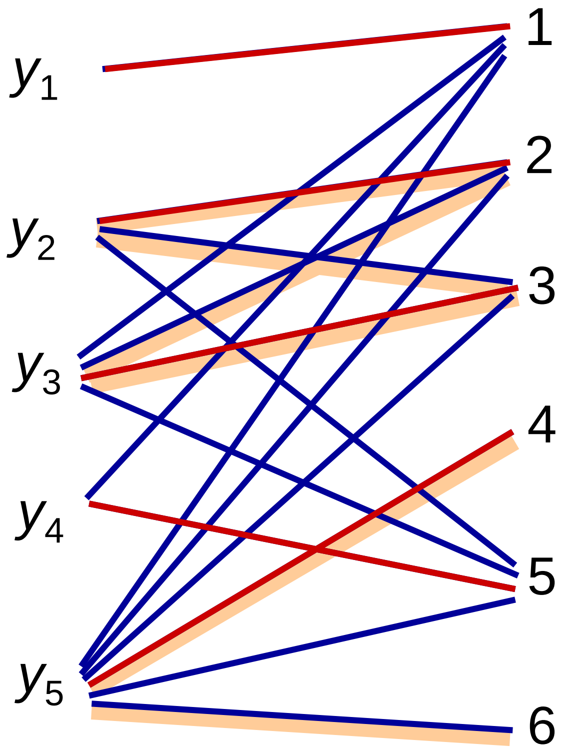


Indicate domains with edges

Find maximum cardinality bipartite matching.

Mark edges in alternating paths that start at an uncovered vertex.

Mark edges in alternating cycles.



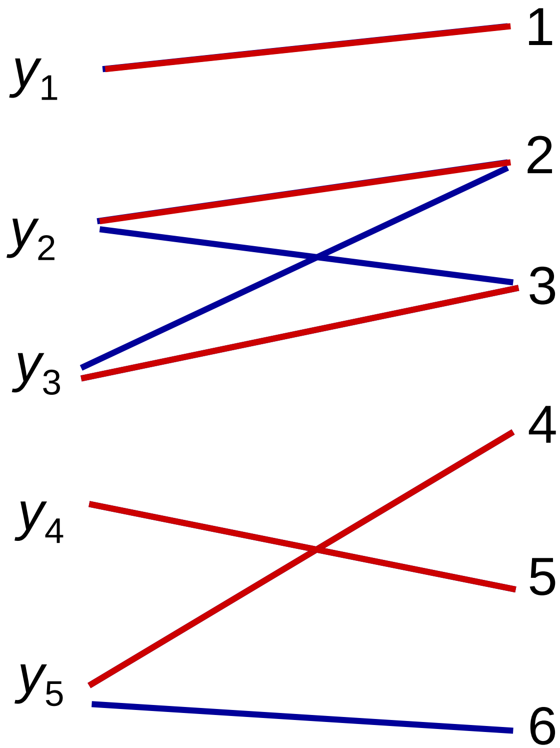
Indicate domains with edges

Find maximum cardinality bipartite matching.

Mark edges in alternating paths that start at an uncovered vertex.

Mark edges in alternating cycles.

Remove unmarked edges not in matching.



Indicate domains with edges

Find maximum cardinality bipartite matching.

Mark edges in alternating paths that start at an uncovered vertex.

Mark edges in alternating cycles.

Remove unmarked edges not in matching.

Filtering for alldiff

Domains have been filtered:

$$
\begin{aligned}
y_1 &\in \{1\} \\
y_2 &\in \{2,3,5\} \\
y_3 &\in \{1,2,3,5\} \\
y_4 &\in \{1,5\} \\
y_5 &\in \{1,2,3,4,5,6\}
\end{aligned}
\quad \longrightarrow \quad
\begin{aligned}
y_1 &\in \{1\} \\
y_2 &\in \{2,3\} \\
y_3 &\in \{2,3\} \\
y_4 &\in \{5\} \\
y_5 &\in \{4,6\}
\end{aligned}
$$

Domain consistency achieved.

Disjunctive scheduling

Consider a disjunctive scheduling constraint:

$$\text{disjunctive}\big((s_1, s_2, s_3, s_5), (p_1, p_2, p_3, p_5)\big)$$

Start time variables

Job j	Release time r_j	Dead-line d_j	Processing time p_{Aj}	p_{Bj}
1	0	10	1	5
2	0	10	3	6
3	2	7	3	7
4	2	10	4	6
5	4	7	2	5

Edge finding for disjunctive scheduling

Consider a disjunctive scheduling constraint:

$$\text{disjunctive}\big((s_1,s_2,s_3,s_5),(p_1,p_2,p_3,p_5)\big)$$

Processing times

Job *j*	*Release time* r_j	*Dead-line* d_j	*Processing time* p_{Aj}	p_{Bj}
1	0	10	1	5
2	0	10	3	6
3	2	7	3	7
4	2	10	4	6
5	4	7	2	5

Edge finding for disjunctive scheduling

Consider a disjunctive scheduling constraint:

$$\text{disjunctive}\left((s_1, s_2, s_3, s_5), (p_1, p_2, p_3, p_5)\right)$$

Job j	*Release time* r_j	*Dead-line* d_j	*Processing time* p_{Aj}	p_{Bj}
1	0	10	1	5
2	0	10	3	6
3	2	7	3	7
4	2	10	4	6
5	4	7	2	5

Variable domains defined by time windows and processing times

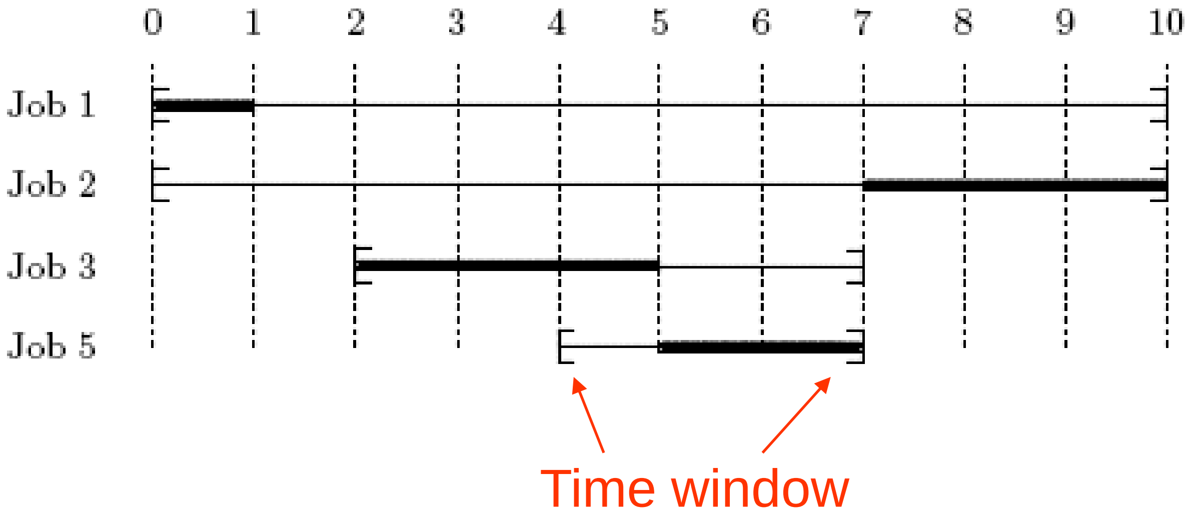
$$s_1 \in [0, 10-1]$$
$$s_2 \in [0, 10-3]$$
$$s_3 \in [2, 7-3]$$
$$s_5 \in [4, 7-2]$$

Edge finding for disjunctive scheduling

Consider a disjunctive scheduling constraint:

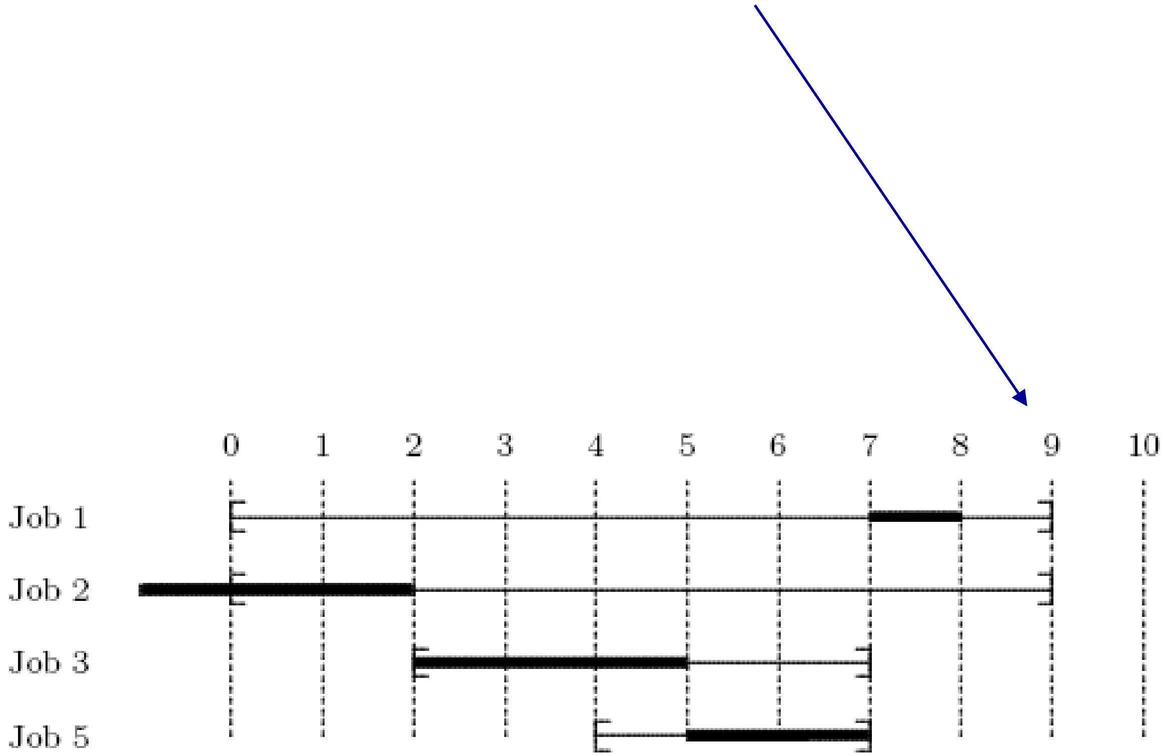
$$\text{disjunctive}\left((s_1, s_2, s_3, s_5), (p_1, p_2, p_3, p_5)\right)$$

A feasible (min makespan) solution:



Edge finding for disjunctive scheduling

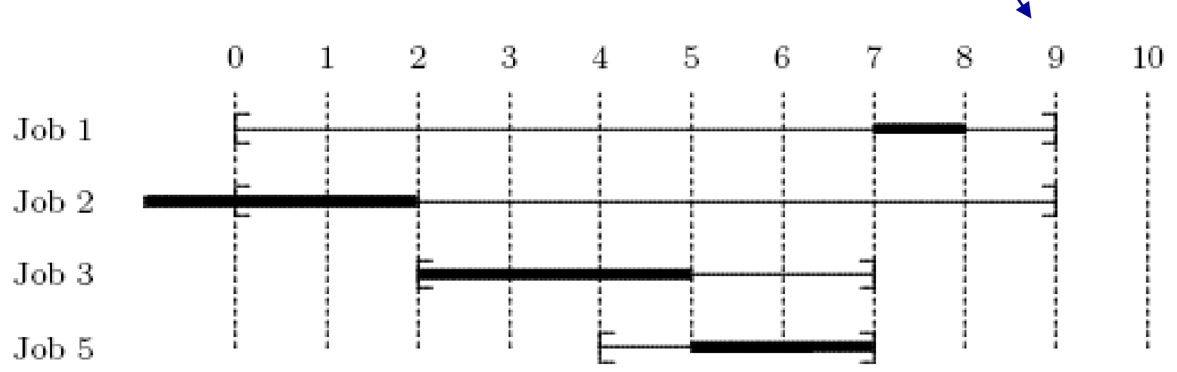
But let's reduce 2 of the deadlines to 9:



Edge finding for disjunctive scheduling

But let's reduce 2 of the deadlines to 9:

We will use edge finding
to prove that there is no
feasible schedule.

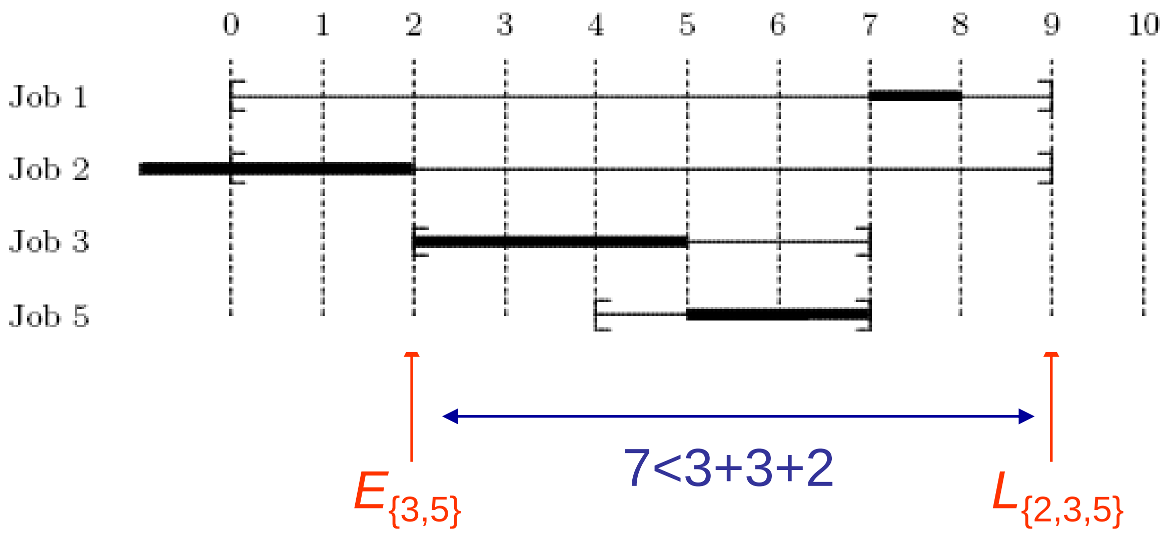


Edge finding for disjunctive scheduling

We can deduce that job 2 must precede jobs 3 and 5:

Because if job 2 is not first, there is not enough time for all 3 jobs within the time windows:

$$L_{\{2,3,5\}} - E_{\{3,5\}} < p_{\{2,3,5\}}$$



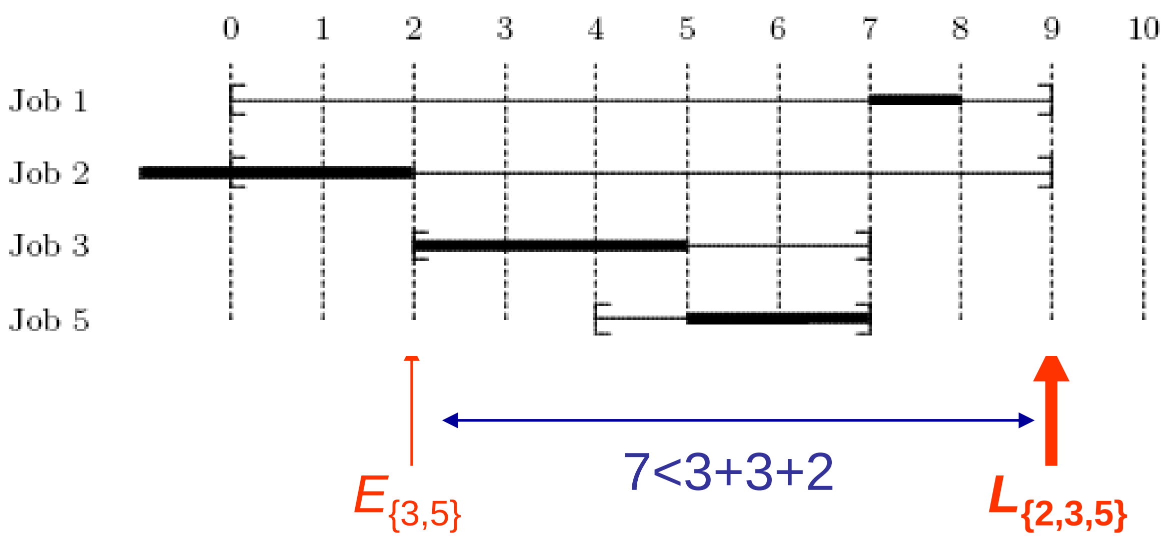
Edge finding for disjunctive scheduling

We can deduce that job 2 must precede jobs 3 and 5:

Because if job 2 is not first, there is not enough time for all 3 jobs within the time windows:

$$\boxed{L_{\{2,3,5\}}} - E_{\{3,5\}} < p_{\{2,3,5\}}$$

Latest deadline



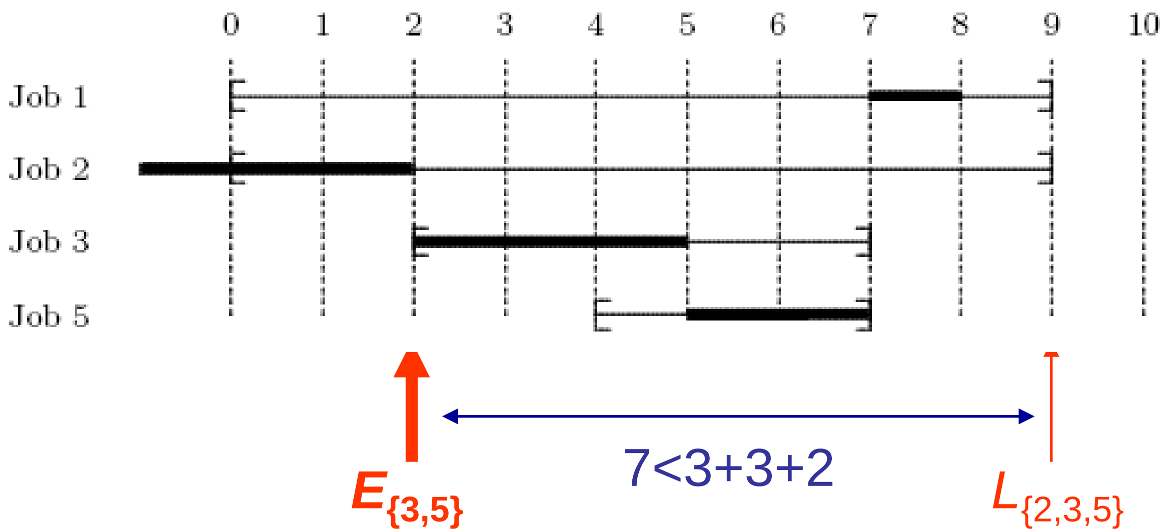
Edge finding for disjunctive scheduling

We can deduce that job 2 must precede jobs 3 and 5:

Because if job 2 is not first, there is not enough time for all 3 jobs within the time windows:

$$L_{\{2,3,5\}} - E_{\{3,5\}} < p_{\{2,3,5\}}$$

Earliest release time



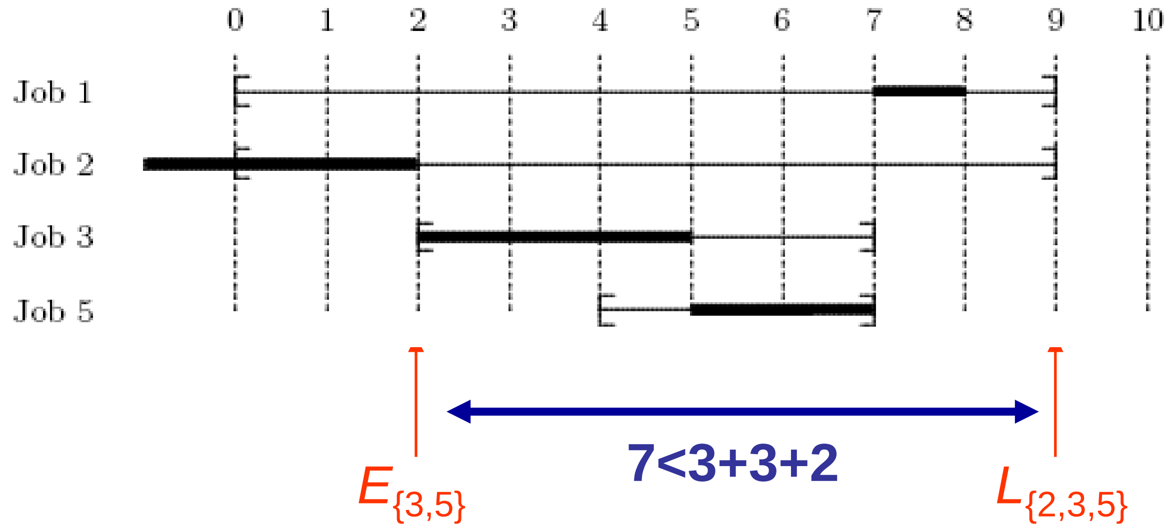
Edge finding for disjunctive scheduling

We can deduce that job 2 must precede jobs 3 and 5:

Because if job 2 is not first, there is not enough time for all 3 jobs within the time windows:

$$L_{\{2,3,5\}} - E_{\{3,5\}} < p_{\{2,3,5\}}$$

Total processing time



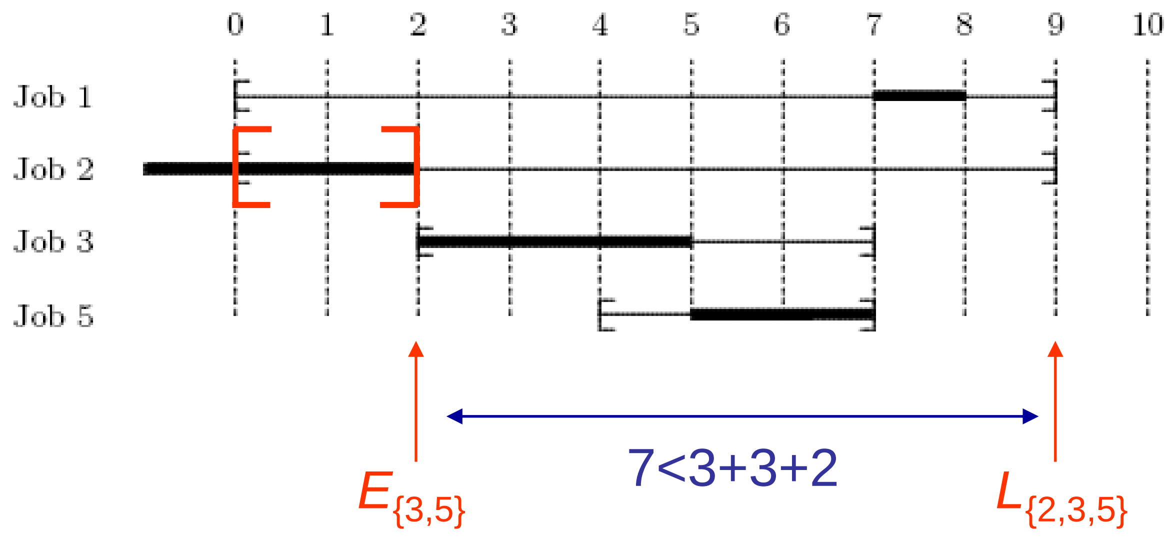
Edge finding for disjunctive scheduling

We can deduce that job 2 must precede jobs 3 and 5:

So we can tighten deadline of job 2 to minimum of

$$L_{\{3\}} - p_{\{3\}} = 4 \qquad L_{\{5\}} - p_{\{5\}} = 5 \qquad L_{\{3,5\}} - p_{\{3,5\}} = 2$$

Since time window of job 2 is now too narrow, there is no feasible schedule.



Edge finding for disjunctive scheduling

In general, we can deduce that job *k* must precede all the jobs in set *J*:

If there is not enough time for all the jobs after the earliest release time of the jobs in *J*

$$L_{J\cup\{k\}} - E_J < p_{J\cup\{k\}} \qquad L_{\{2,3,5\}} - E_{\{3,5\}} < p_{\{2,3,5\}}$$

Edge finding for disjunctive scheduling

In general, we can deduce that job *k* must precede all the jobs in set *J*:

If there is not enough time for all the jobs after the earliest release time of the jobs in *J*

$$L_{J\cup\{k\}} - E_J < p_{J\cup\{k\}} \qquad L_{\{2,3,5\}} - E_{\{3,5\}} < p_{\{2,3,5\}}$$

Now we can tighten the deadline for job *k* to:

$$\min_{J'\subset J}\left\{L_{J'} - p_{J'}\right\} \qquad L_{\{3,5\}} - p_{\{3,5\}} = 2$$

Edge finding for disjunctive scheduling

There is a symmetric rule:

If there is not enough time for all the jobs before the latest deadline of the jobs in *J:*

$$L_J - E_{J\cup\{k\}} < p_{J\cup\{k\}}$$

Now we can tighten the release date for job *k* to*:*

$$\max_{J'\subset J}\left\{E_{J'} + p_{J'}\right\}$$

Edge finding for disjunctive scheduling

Problem: how can we avoid enumerating all subsets *J* of jobs to find edges?

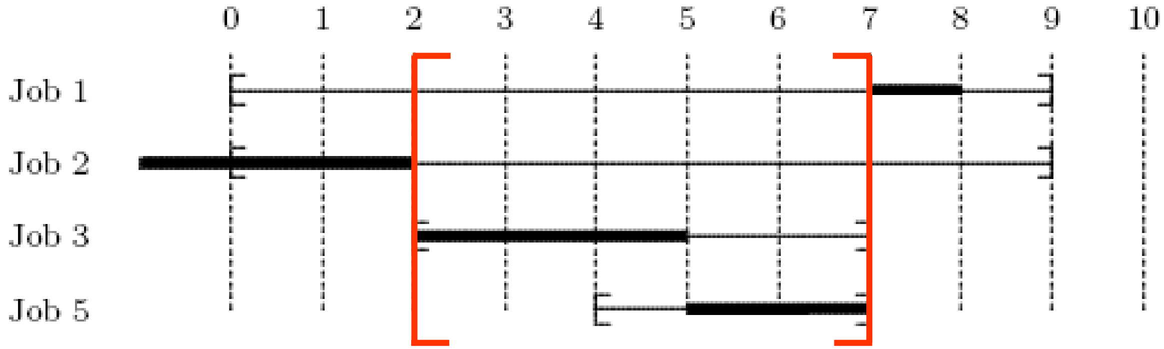
$$L_{J\cup\{k\}} - E_J < p_{J\cup\{k\}}$$

…and all subsets *J′* of *J* to tighten the bounds?

$$\min_{J'\subset J}\left\{L_{J'} - p_{J'}\right\}$$

Edge finding for disjunctive scheduling

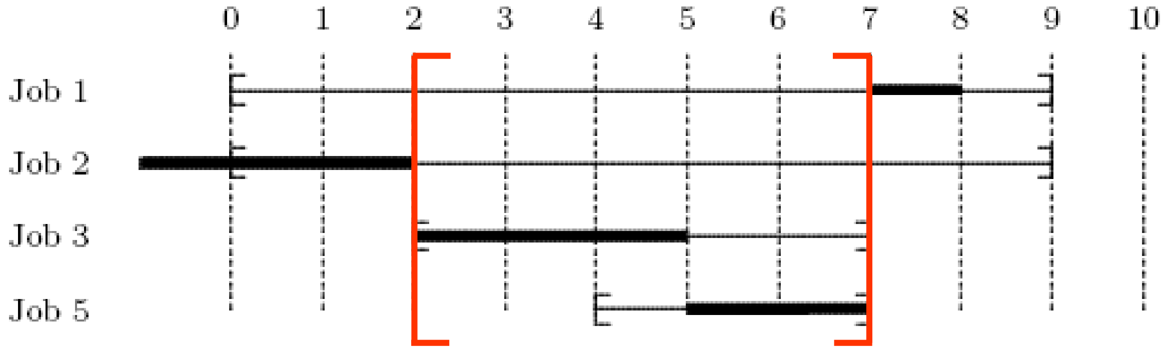
Key result: We only have to consider sets *J* whose time windows lie within some interval.



e.g., $J = \{3,5\}$

Edge finding for disjunctive scheduling

Key result: We only have to consider sets *J* whose time windows lie within some interval.



e.g., $J = \{3,5\}$

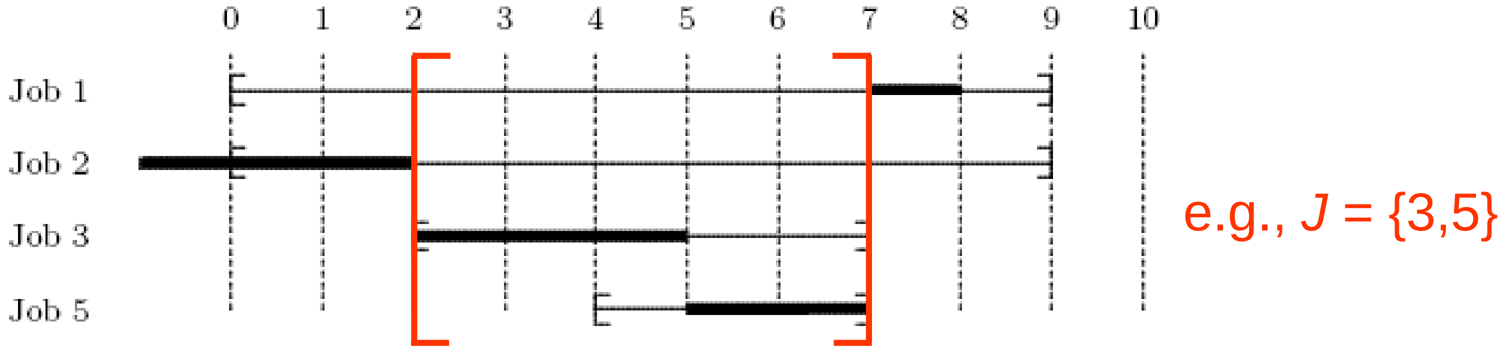
Removing a job from those within an interval only weakens the test

$$L_{J \cup \{k\}} - E_J < p_{J \cup \{k\}}$$

There are a polynomial number of intervals defined by release times and deadlines.

Edge finding for disjunctive scheduling

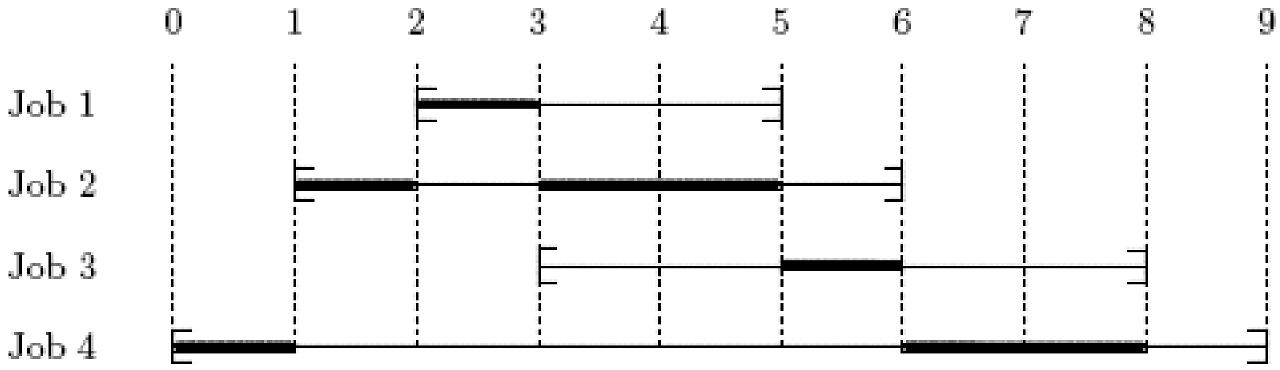
Key result: We only have to consider sets *J* whose time windows lie within some interval.



Note: Edge finding does not achieve bounds consistency, which is an NP-hard problem.

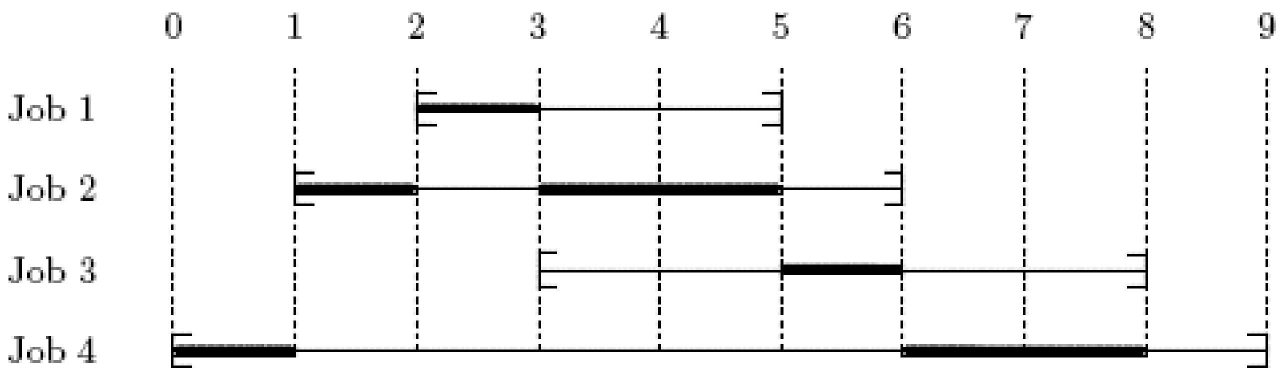
Edge finding for disjunctive scheduling

One $O(n^2)$ algorithm is based on the Jackson pre-emptive schedule (JPS). Using a different example, the JPS is:



Edge finding for disjunctive scheduling

One $O(n^2)$ algorithm is based on the Jackson pre-emptive schedule (JPS). Using a different example, the JPS is:



For each job i

Scan jobs $k \in J_i$ in decreasing order of L_k

Select first k for which $L_k - E_i < p_i + \bar{p}_{J_{ik}}$

Conclude that i □ J_{ik}

Update E_i to $\text{JPS}(i,k)$

Jobs unfinished at time E_i in JPS

Total remaining processing time in JPS of jobs in J_{ik}

Jobs $j \neq i$ in J_i with $L_j \leq L_k$

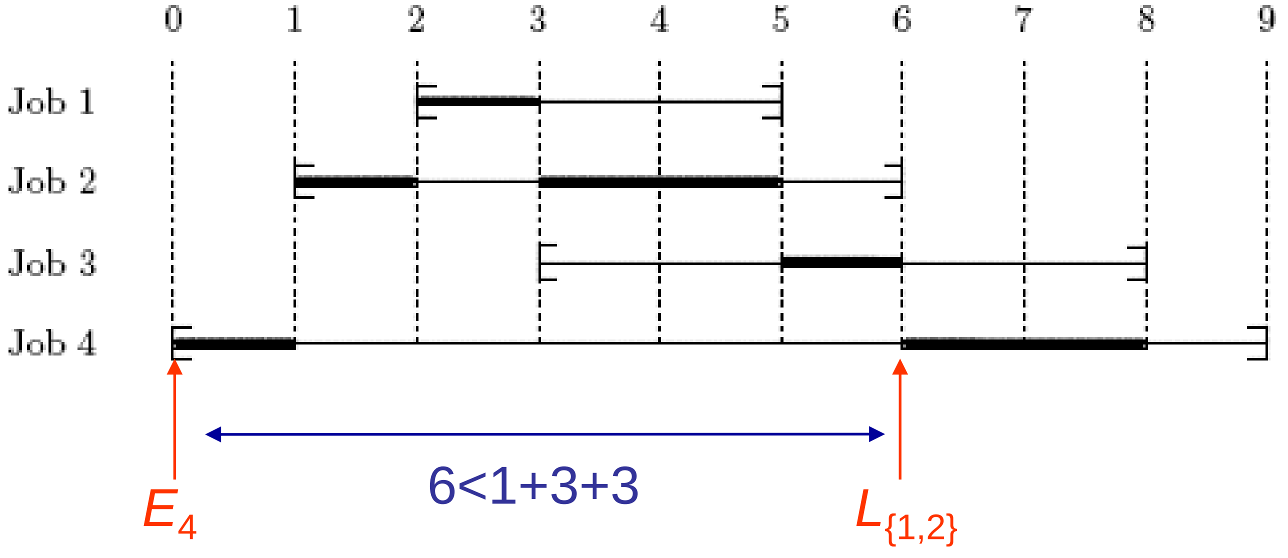
Latest completion time in JPS of jobs in J_{ik}

Not-first/not-last rules

We can deduce that job 4 cannot precede jobs 1 and 2:

Because if job 4 is first, there is too little time to complete the jobs before the later deadline of jobs 1 and 2:

$$L_{\{1,2\}} - E_4 < p_1 + p_2 + p_4$$

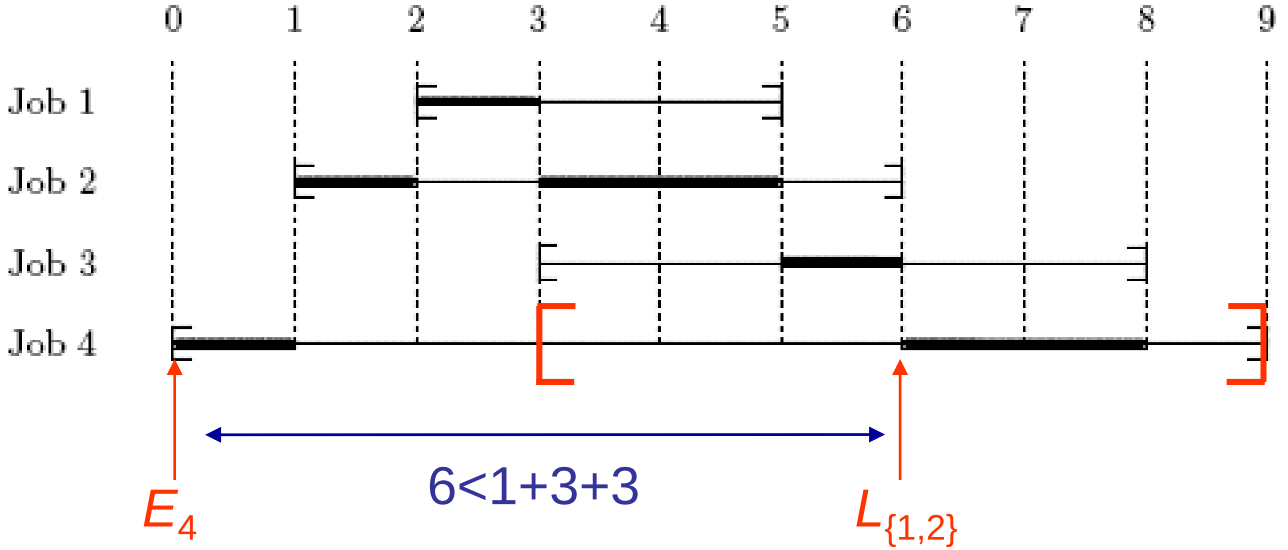


Not-first/not-last rules

We can deduce that job 4 cannot precede jobs 1 and 2:

Now we can tighten the release time of job 4 to minimum of:

$$E_1 + p_1 = 3 \qquad E_2 + p_2 = 4$$



Not-first/not-last rules

In general, we can deduce that job *k* cannot precede all the jobs in *J*:

if there is too little time after release time of job *k* to complete all jobs before the latest deadline in *J*:

$$L_J - E_k < p_J$$

Now we can update E_i to

$$\min_{j \in J}\left\{E_j + p_j\right\}$$

Not-first/not-last rules

In general, we can deduce that job *k* cannot precede all the jobs in *J*:

if there is too little time after release time of job *k* to complete all jobs before the latest deadline in *J*:

$$L_J - E_k < p_J$$

Now we can update E_i to

$$\min_{j \in J} \{E_j + p_j\}$$

There is a symmetric not-last rule.

The rules can be applied in polynomial time, although an efficient algorithm is quite complicated.

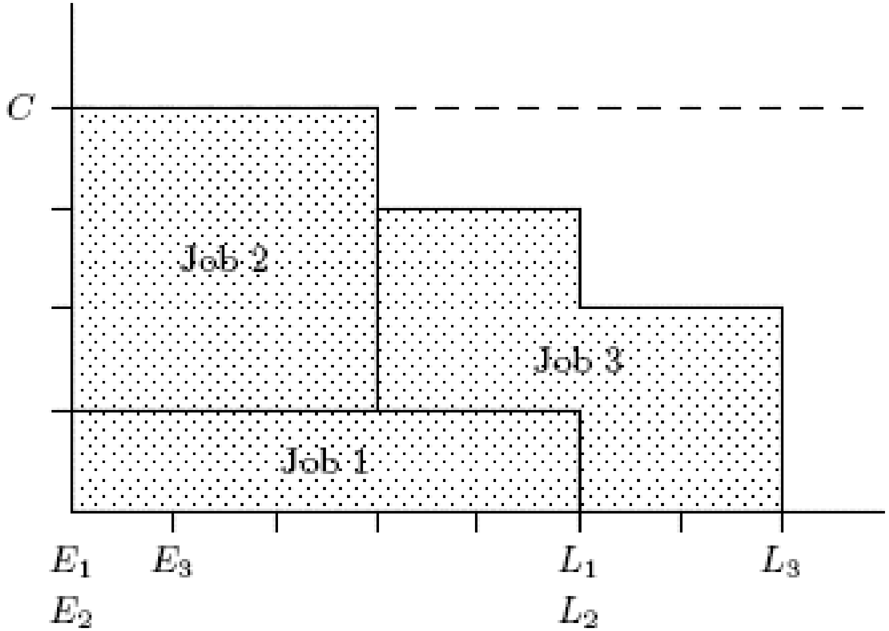
Cumulative scheduling

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:

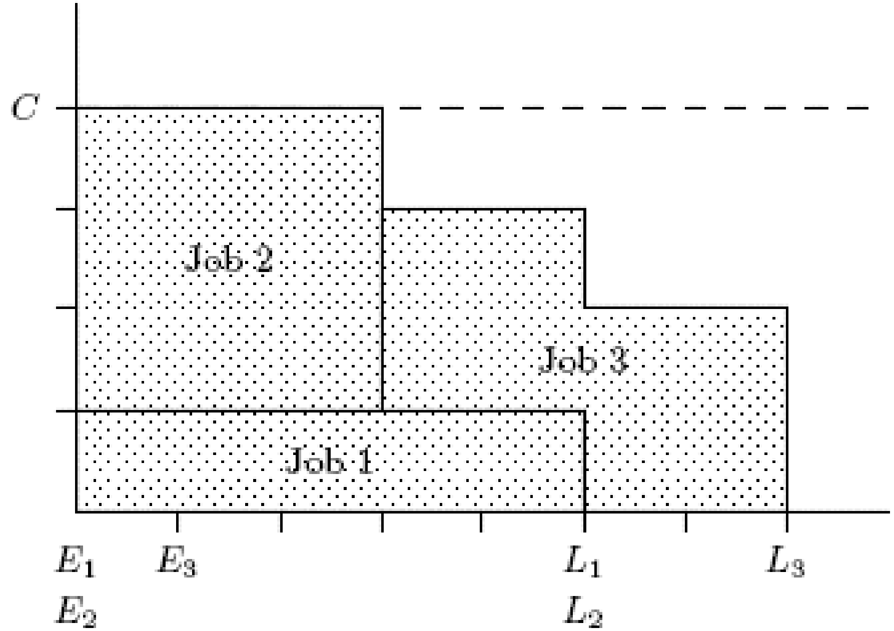


Edge finding for cumulative scheduling

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:

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



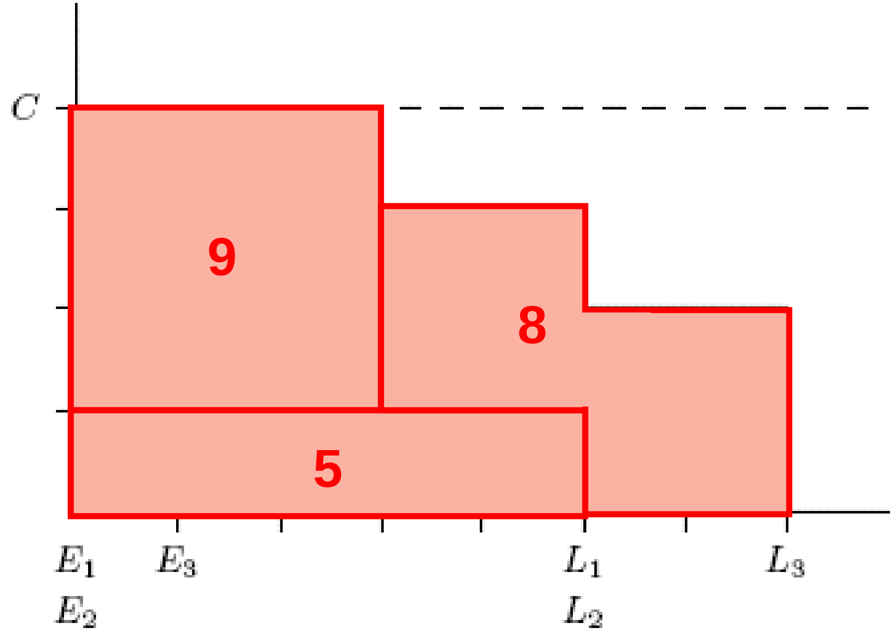
Edge finding for cumulative scheduling

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

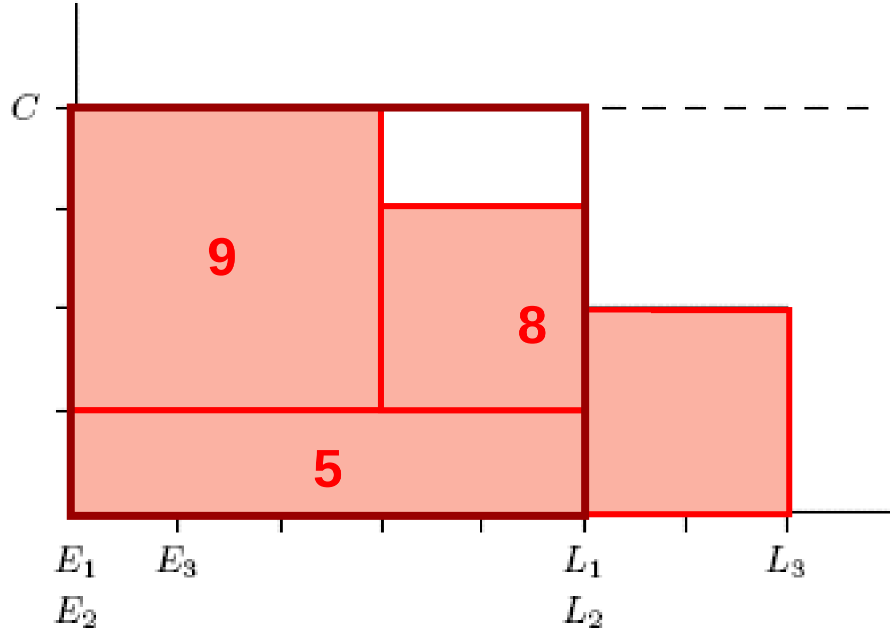
We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

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



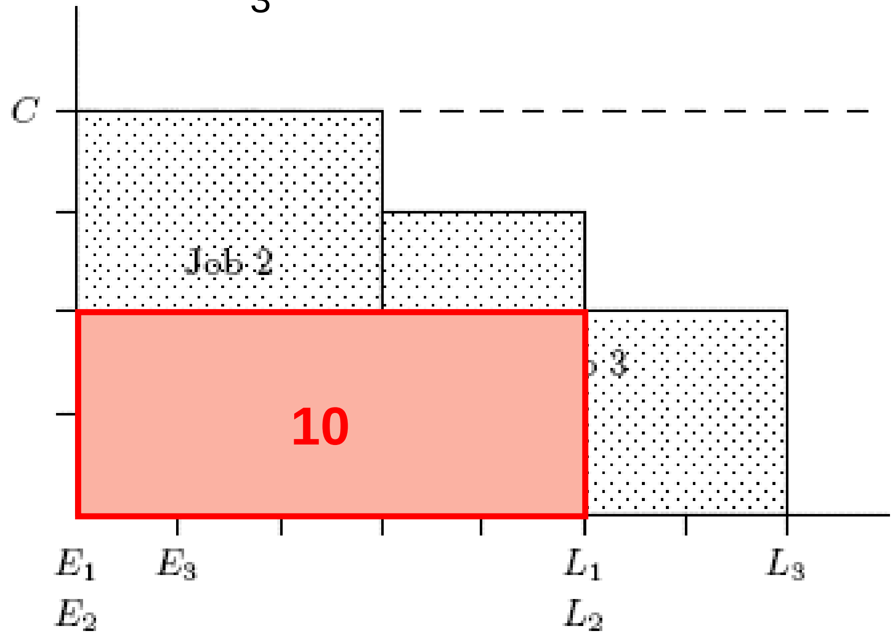
Edge finding for cumulative scheduling

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

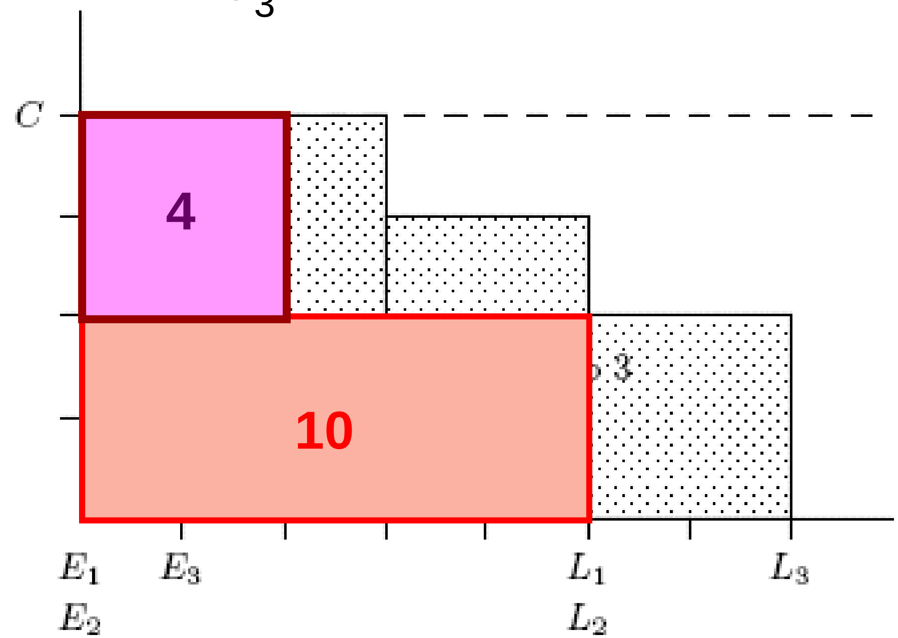
We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

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

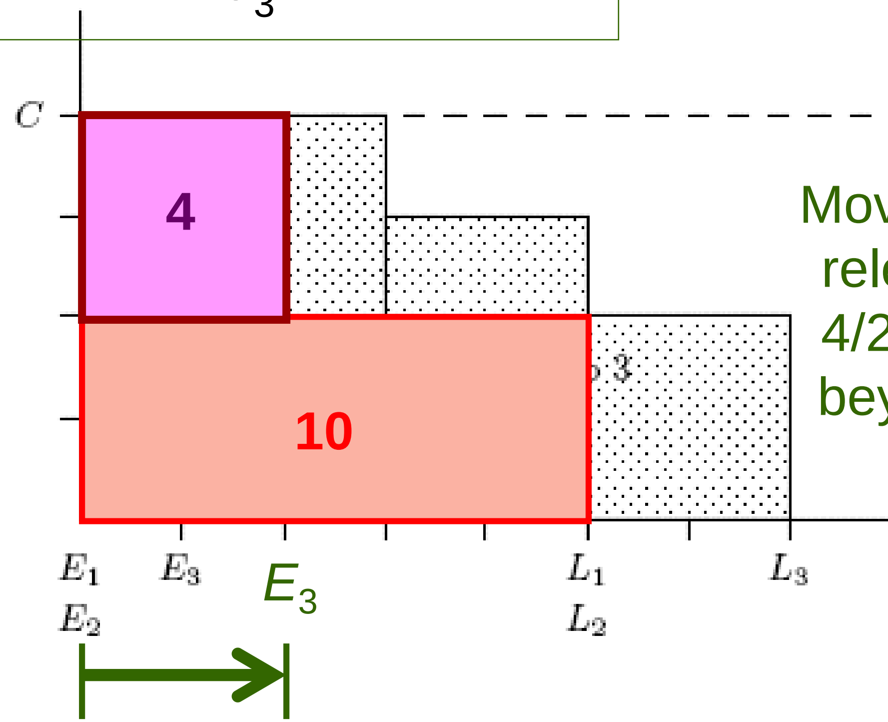
We can deduce that job 3 must finish after the others finish: $3 > \{1,2\}$

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

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

Other propagation rules for cumulative scheduling

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

CP-based Branch and Price

Basic Idea

Example: Airline Crew Scheduling

Motivation

- **Branch and price** allows solution of integer programming problems with a huge number of variables.
- The problem is **solved by branching**, like a normal IP. The difference lies in how the LP relaxation is solved.
- Variables are added to the LP relaxation **only as needed**.
- Variables are **priced** to find which ones should be added.
- **CP** is useful for solving the **pricing problem**, particularly when constraints are complex.
- **CP-based branch and price** has been successfully applied to airline crew scheduling, transit scheduling, and other transportation-related problems.

Basic Idea

Suppose the LP relaxation of an integer programming problem has a huge number of variables:

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

We will solve a **restricted master problem**, which has a small subset of the variables:

$$\min \; \sum_{j \in J} c_j x_j$$
$$\sum_{j \in J} A_j x_j = b \quad (\lambda)$$
$$x_j \geq 0$$

Column j of A

Adding x_k to the problem would improve the solution if x_k has a negative reduced cost:

$$r_k = c_k - \lambda A_k < 0$$

Basic Idea

Adding x_k to the problem would improve the solution if x_k has a negative reduced cost:

$$r_k = c_k - \lambda A_k < 0$$

Computing the reduced cost of x_k is known as **pricing** x_k.

Cost of column *y*

So we solve the pricing problem:

$$\min \; c_y - \lambda y$$
$$y \text{ is a column of } A$$

If the solution y^* satisfies $c_{y^*} - \lambda y^* < 0$, then we can add column y to the restricted master problem.

Basic Idea

The pricing problem $\max \; \lambda y$

y is a column of A

need not be solved to optimality, so long as we find a column with negative reduced cost.

However, when we can no longer find an improving column, we solved the pricing problem to optimality to make sure we have the optimal solution of the LP.

If we can state constraints that the columns of *A* must satisfy, CP may be a good way to solve the pricing problem.

Example: Airline Crew Scheduling

We want to assign crew members to flights to minimize cost while covering the flights and observing complex work rules.

Flight data

j	s_j	f_j
1	0	3
2	1	3
3	5	8
4	6	9
5	10	12
6	14	16

Start time (s_j) Finish time (f_j)

A **roster** is the sequence of flights assigned to a single crew member.

The gap between two consecutive flights in a roster must be from 2 to 3 hours. Total flight time for a roster must be between 6 and 10 hours.

For example,

flight 1 cannot immediately precede 6

flight 4 cannot immediately precede 5.

The possible rosters are:

(1,3,5), (1,4,6), (2,3,5), (2,4,6)

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1	2	3	4
(1,3,5),	(1,4,6),	(2,3,5),	(2,4,6)

The LP relaxation of the problem is:

Cost of assigning crew member 1 to roster 2

$$\min\ z$$

$$\begin{bmatrix} 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$x_{ik} \geq 0,\ \text{all } i, k$$

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1	2	3	4
(1,3,5),	(1,4,6),	(2,3,5),	(2,4,6)

The LP relaxation of the problem is:

Cost of assigning crew member 1 to roster 2

$$\min\ z$$

$$\begin{bmatrix} 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$x_{ik} \geq 0, \text{ all } i, k$$

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 1.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1	2	3	4
(1,3,5),	(1,4,6),	(2,3,5),	(2,4,6)

The LP relaxation of the problem is:

Cost of assigning crew member 1 to roster 2

$$\min\ z$$

$$\begin{bmatrix} 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$x_{ik} \geq 0, \ \text{all } i, k$$

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 2.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1	2	3	4
(1,3,5),	(1,4,6),	(2,3,5),	(2,4,6)

The LP relaxation of the problem is:

$$\min\ z$$

$$\begin{bmatrix} 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$x_{ik} \geq 0, \text{ all } i, k$$

Cost of assigning crew member 1 to roster 2

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 3.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1	2	3	4
(1,3,5),	(1,4,6),	(2,3,5),	(2,4,6)

The LP relaxation of the problem is:

Cost of assigning crew member 1 to roster 2

$$\min z$$

$$\begin{bmatrix} 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$x_{ik} \geq 0, \text{ all } i, k$$

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 4.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1 2 3 4

(1,3,5), (1,4,6), (2,3,5), (2,4,6)

The LP relaxation of the problem is:

Cost of assigning crew member 1 to roster 2

$$\min \ z$$

$$\begin{bmatrix} 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$x_{ik} \geq 0, \ \text{all } i, k$$

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 5.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1 2 3 4

(1,3,5), (1,4,6), (2,3,5), (2,4,6)

The LP relaxation of the problem is:

Cost of assigning crew member 1 to roster 2

$$\min\ z$$

$$\begin{bmatrix} 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$x_{ik} \geq 0, \ \text{all } i, k$$

= 1 if we assign crew member 1 to roster 2, = 0 otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

Rosters that cover flight 6.

Airline Crew Scheduling

There are 2 crew members, and the possible rosters are:

1	2	3	4
(1,3,5),	(1,4,6),	(2,3,5),	(2,4,6)

The LP relaxation of the problem is:

Cost c_{12} of assigning crew member 1 to roster 2

$$\min\ z$$

$$\begin{bmatrix} 10 & 12 & 7 & 13 & 9 & 11 & 6 & 12 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$x_{ik} \geq 0, \text{ all } i, k$$

$= 1$ if we assign crew member 1 to roster 2, $= 0$ otherwise.

Each crew member is assigned to exactly 1 roster.

Each flight is assigned at least 1 crew member.

In a real problem, there can be **millions** of rosters.

Airline Crew Scheduling

We start by solving the problem with a subset of the columns:

$$\min\ z$$

$$\begin{bmatrix} 10 & 13 & 9 & 12 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{14} \\ x_{21} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad \begin{matrix} (10) & u_1 \\ (9) & u_2 \\ (0) & v_1 \\ (0) & v_2 \\ (0) & v_3 \\ (0) & v_4 \\ (0) & v_5 \\ (3) & v_6 \end{matrix}$$

$$x_{ik} \geq 0,\ \text{all } i, k$$

Optimal dual solution

Airline Crew Scheduling

We start by solving the problem with a subset of the columns:

$$\min\ z$$

$$\begin{bmatrix} 10 & 13 & 9 & 12 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{14} \\ x_{21} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad \begin{matrix} \\ (10) \\ (9) \\ (0) \\ (0) \\ (0) \\ (0) \\ (0) \\ (3) \end{matrix} \quad \begin{matrix} \\ u_1 \\ u_2 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix}$$

$$x_{ik} \geq 0, \text{ all } i, k$$

Dual variables

Airline Crew Scheduling

We start by solving the problem with a subset of the columns:

$$\min \ z$$

$$\begin{bmatrix} 10 & 13 & 9 & 12 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{14} \\ x_{21} \\ x_{24} \end{bmatrix} \begin{matrix} = \\ = \\ = \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \\ \geq \end{matrix} \begin{bmatrix} z \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad \begin{matrix} \\ (10) \\ (9) \\ (0) \\ (0) \\ (0) \\ (0) \\ (0) \\ (3) \end{matrix}$$

$$x_{ik} \geq 0, \ \text{all } i, k$$

Dual variables

u_1, u_2, v_1, v_2, v_3, v_4, v_5, v_6

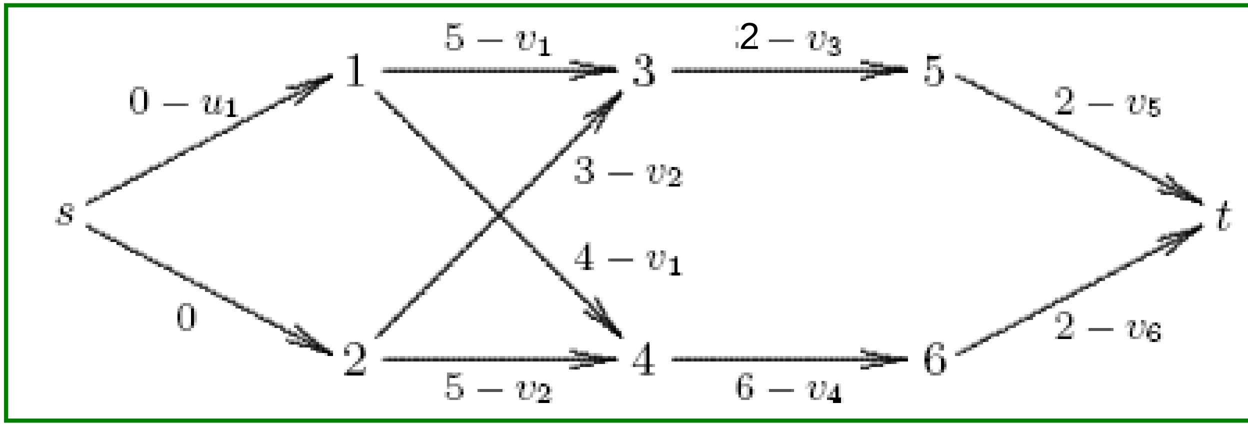
The reduced cost of an excluded roster *k* for crew member *i* is

$$c_{ik} - u_i - \sum_{j \text{ in roster k}} v_j$$

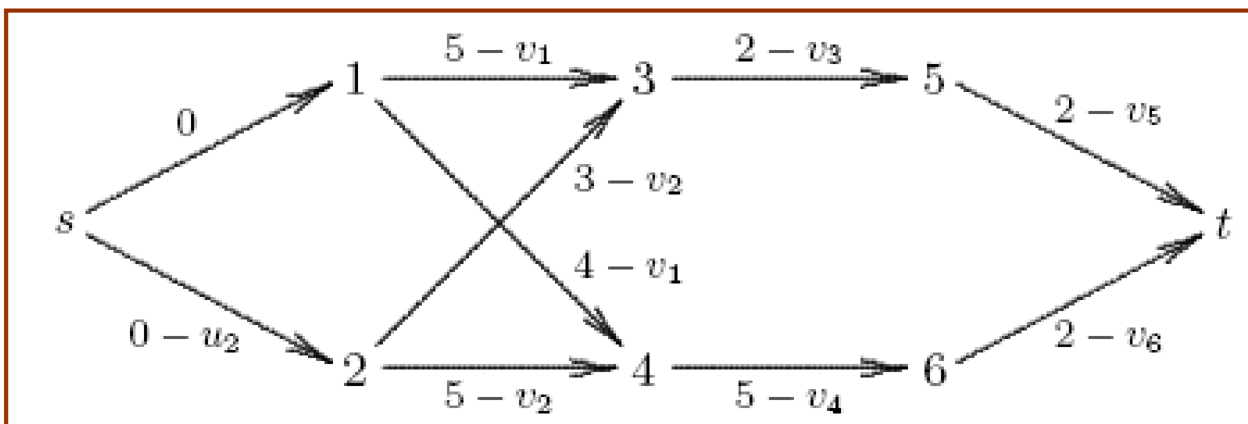
We will formulate the pricing problem as a shortest path problem.

Pricing problem

Crew member 1



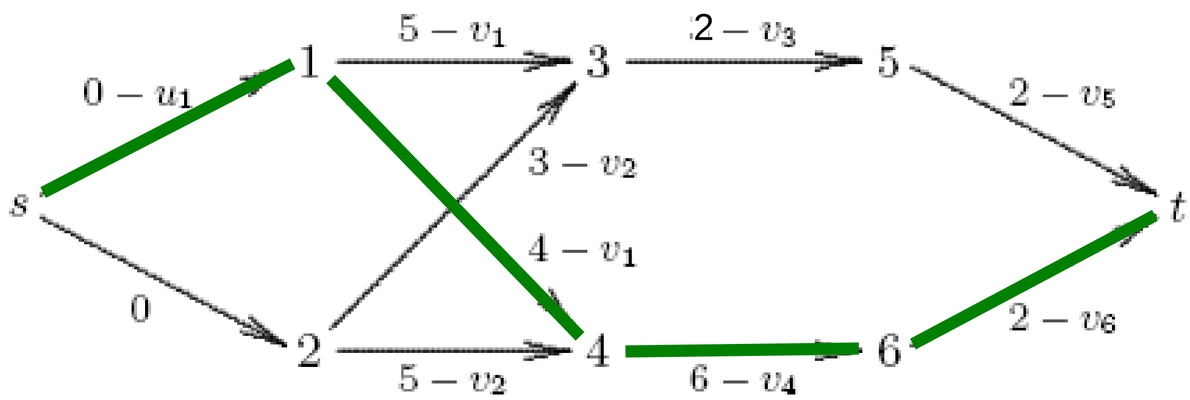
Crew member 2



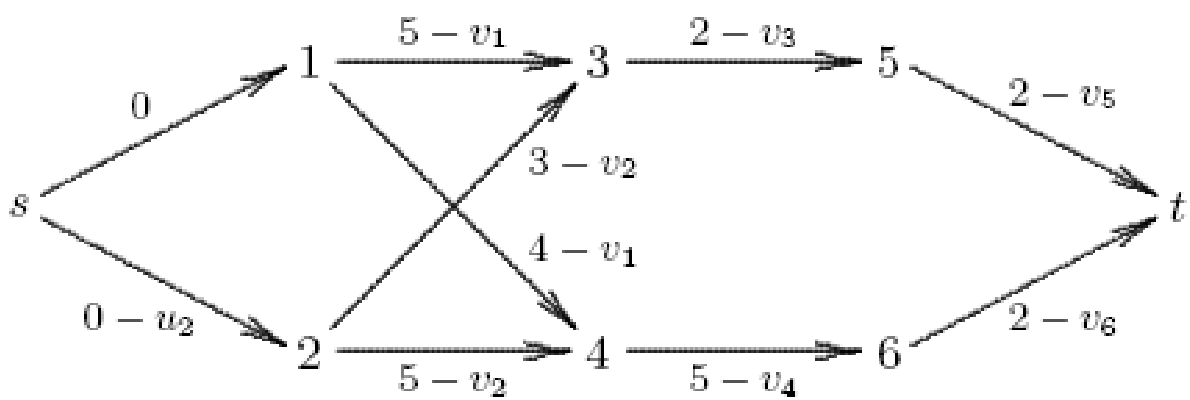
Pricing problem

Each s-t path corresponds to a roster, provided the flight time is within bounds.

Crew member 1



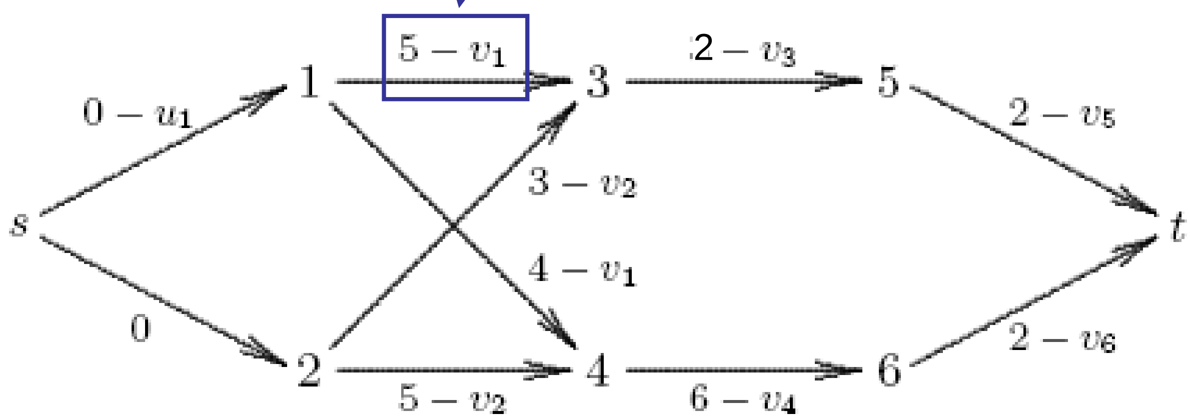
Crew member 2



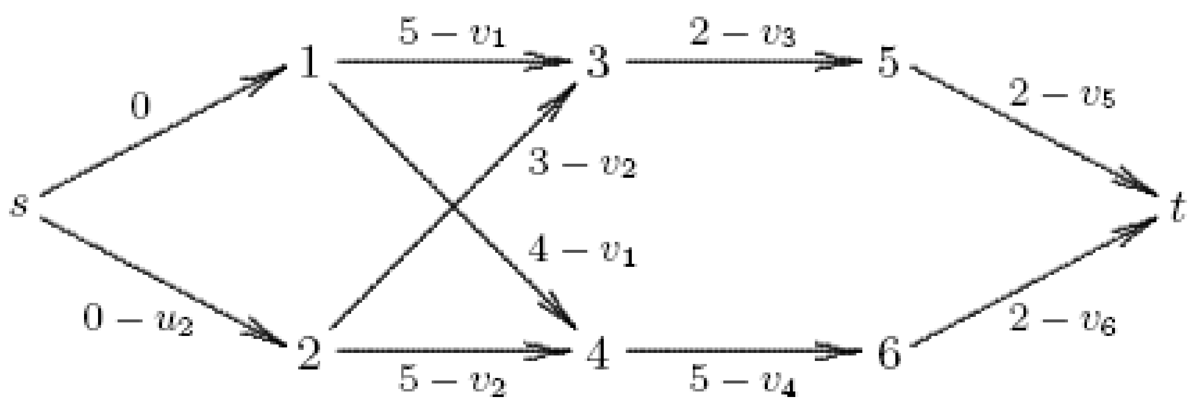
Pricing problem

Cost of flight 3 if it immediately follows flight 1, offset by dual multiplier for flight 1

Crew member 1



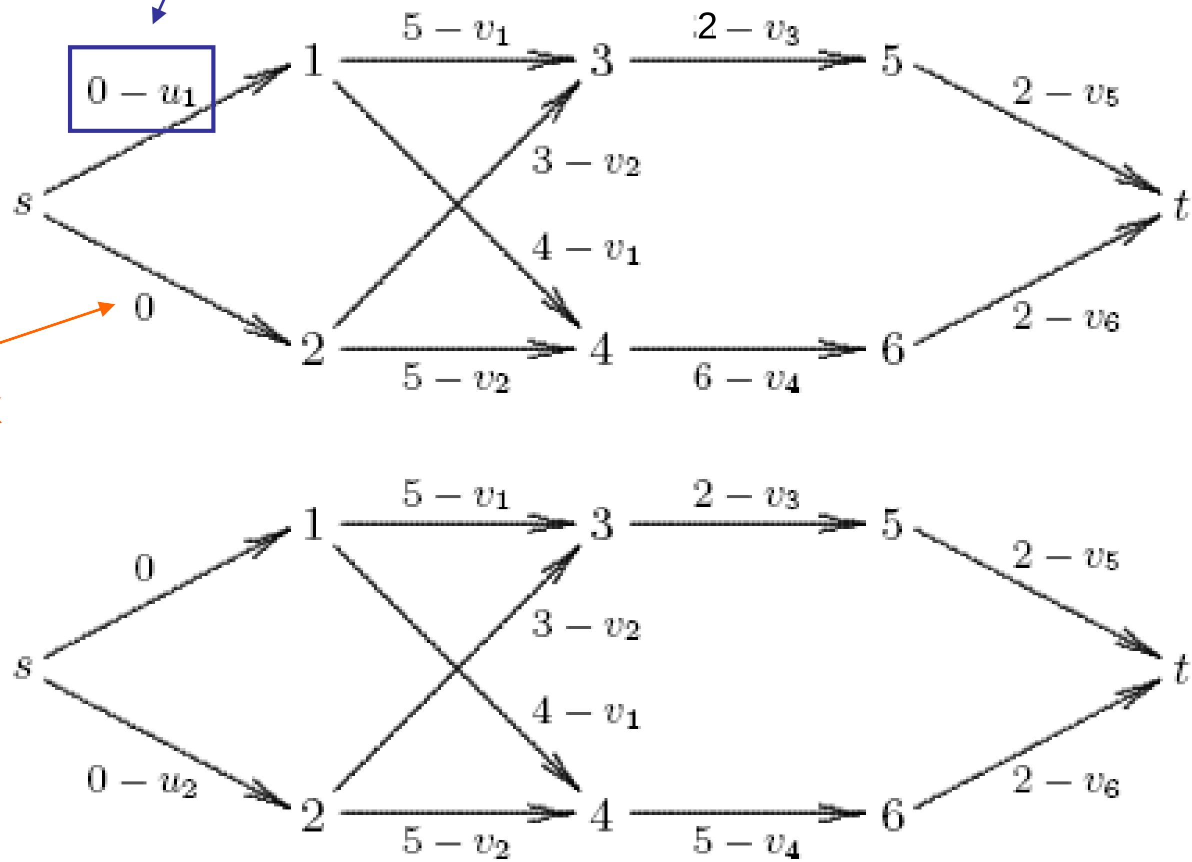
Crew member 2



Pricing problem

Cost of transferring from home to flight 1, offset by dual multiplier for crew member 1

Crew member 1



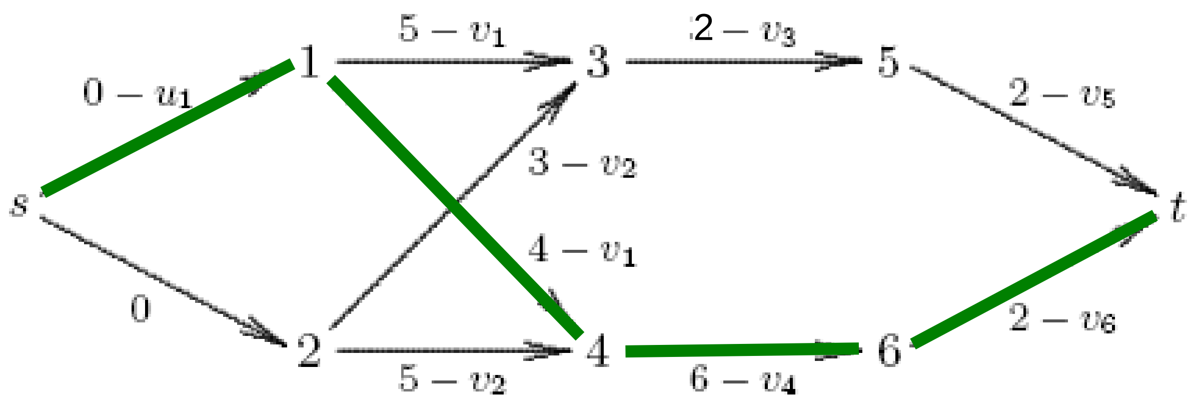
Dual multiplier omitted to break symmetry

Crew member 2

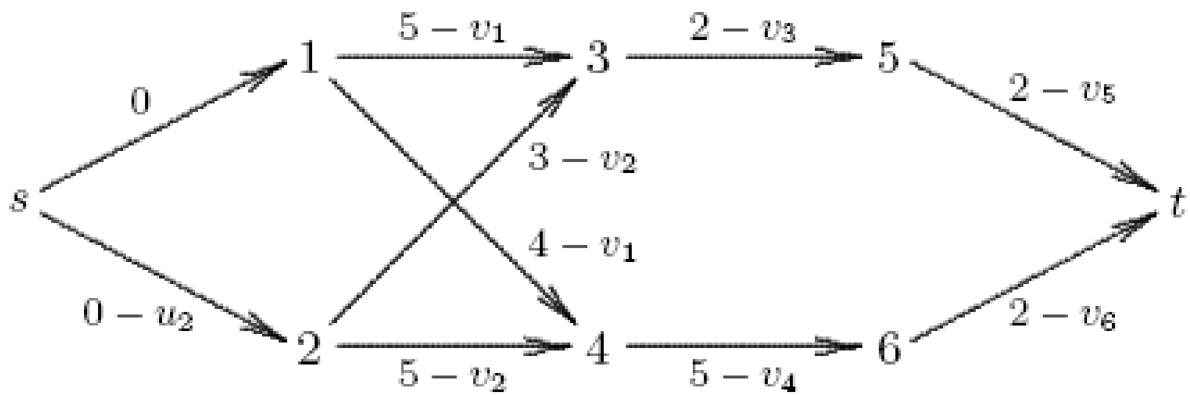
Pricing problem

Length of a path is reduced cost of the corresponding roster.

Crew member 1



Crew member 2

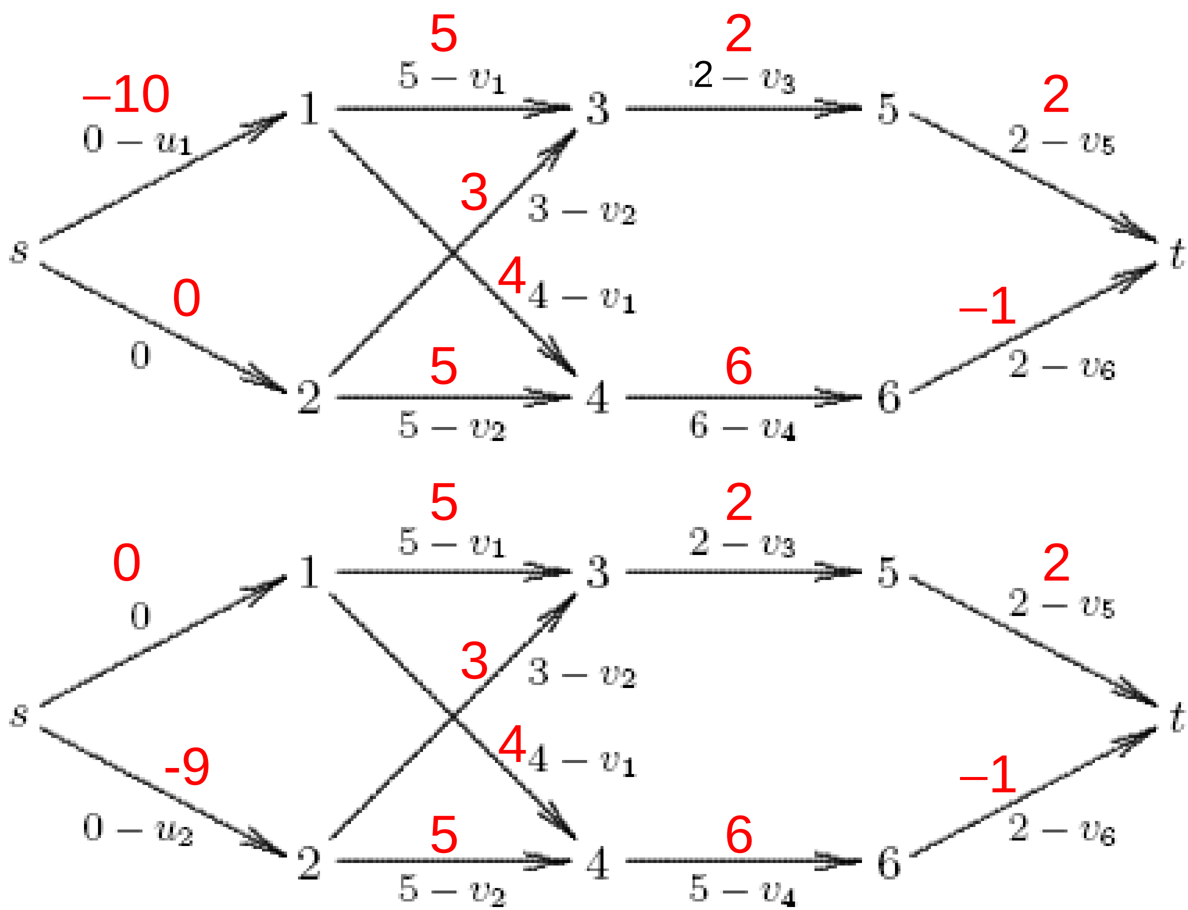


Pricing problem

Arc lengths using dual solution of LP relaxation

Crew member 1

Crew member 2

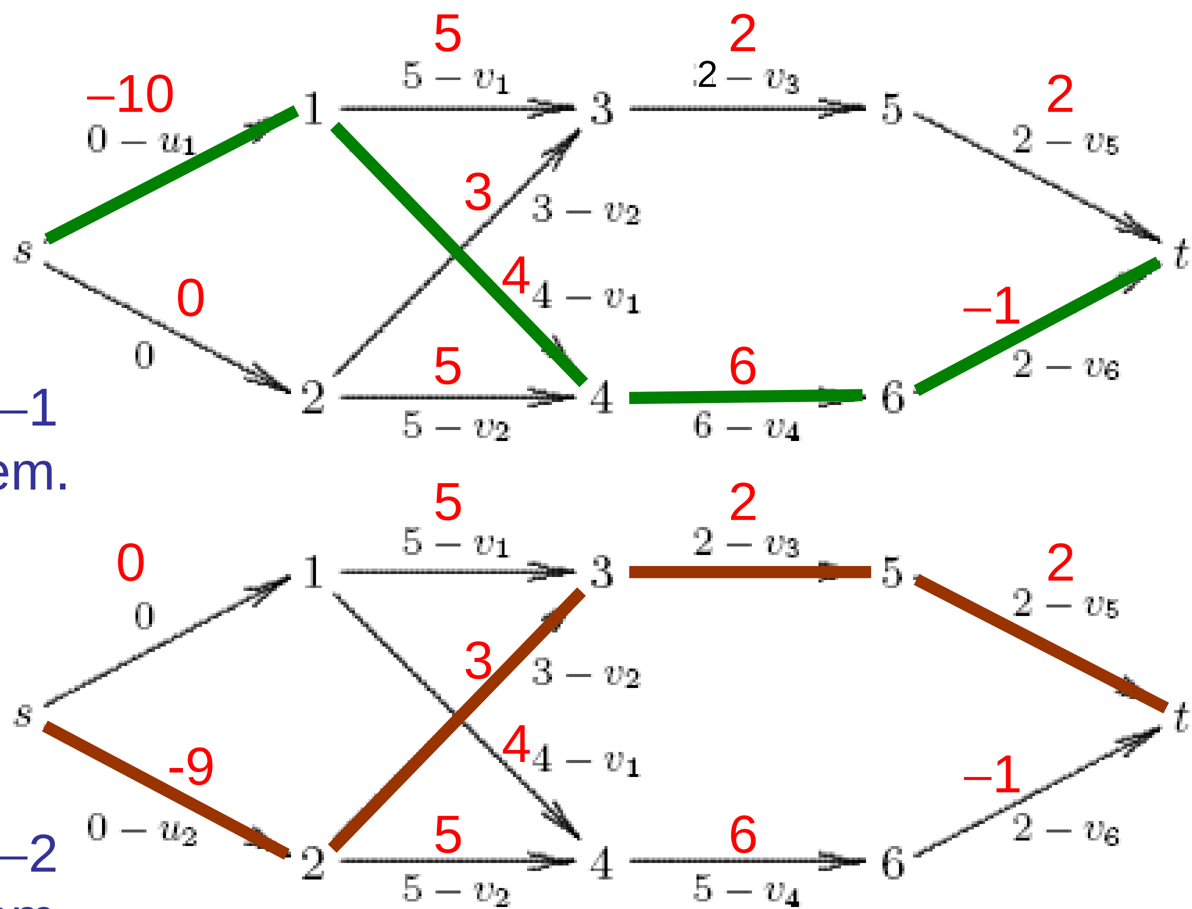


Pricing problem

Solution of shortest path problems

Crew member 1

Reduced cost = –1
Add x_{12} to problem.



Crew member 2

Reduced cost = –2
Add x_{23} to problem.

After x_{12} and x_{23} are added to the problem, no remaining variable has negative reduced cost.

Pricing problem

The shortest path problem cannot be solved by traditional shortest path algorithms, due to the bounds on total duration of flights.

It **can** be solved by CP:

Set of flights assigned to crew member *i*

Path length

Graph

Path global constraint → $\text{Path}(X_i, z_i, G)$, all flights i

Setsum global constraint → $T_{\min} \le \sum_{j \in X_i} (f_j - s_j) \le T_{\max}$

$X_i \subset \{\text{flights}\},\ \ z_i < 0,\ \text{all } i$

Duration of flight *j*

Benders Decomposition

Logic-Based Benders Decomposition

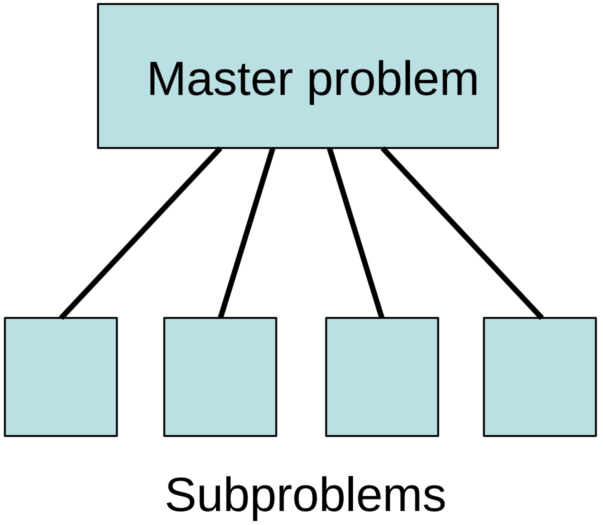
Some Applications

Example: Machine Scheduling

Application: Home Health Care

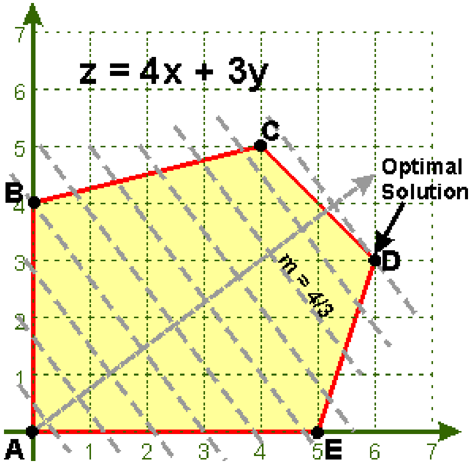
Benders Decomposition

- Benders decomposition is a classical strategy that does not sacrifice overall optimality.
  - Separates the problem into a master problem and multiple subproblems.
    - Variables are partitioned between master and subproblems.
    - Exploits the fact that the problem may radically simplify when the master problem variables are fixed to a set of values.



Benders Decomposition

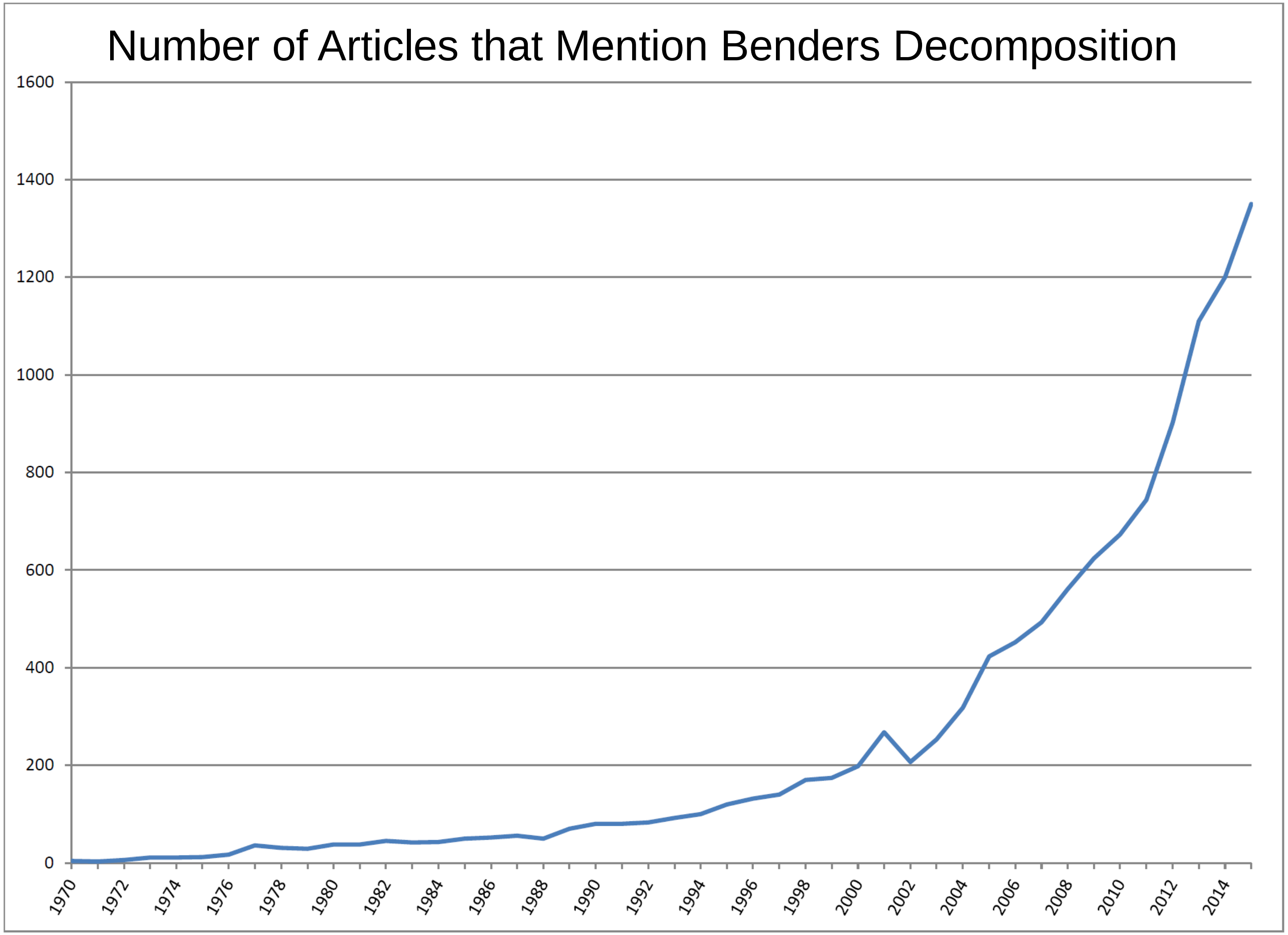
- But classical Benders decomposition has a serious limitation.
  - The subproblems must be linear programming problems.
    - Or continuous nonlinear programming problems.
    - The linear programming dual provides the Benders cuts.



Logic-Based Benders

- Logic-based Benders decomposition attempts to overcome this limitation.
  - The subproblem can be any optimization/feasibility problem, such as a CP problem
    - The Benders cuts are obtained from an inference dual.
  - Speedup over state of the art can be several orders of magnitude.
  - Yet the Benders cuts must be designed specifically for every class of problems.

Logic-Based Benders



Source: Google Scholar

Logic-Based Benders

- Logic-based Benders decomposition solves a problem of the form

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

  – Where the problem simplifies when *x* is fixed to a specific value.

Logic-Based Benders

- Decompose problem into master and subproblem.
  - Subproblem is obtained by fixing x to solution value in master problem.

Master problem

$$\min z$$
$$z \geq g_k(x) \quad \text{(Benders cuts)}$$
$$x \in D_x$$

Minimize cost z subject to bounds given by Benders cuts, obtained from values of x attempted in previous iterations k.

→ Trial value $\bar{x}$ that solves master

← Benders cut $z \geq g_k(x)$

Subproblem

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

Obtain proof of optimality (solution of inference dual). Use same proof to deduce cost bounds for other assignments, yielding Benders cut.

Logic-Based Benders

- Iterate until master problem value equals best subproblem value so far.
  - This yields optimal solution.

Master problem

$$\min z$$
$$z \geq g_k(x) \quad \text{(Benders cuts)}$$
$$x \in D_x$$

Minimize cost z subject to bounds given by Benders cuts, obtained from values of x attempted in previous iterations k.

Trial value $\bar{x}$ that solves master →

← Benders cut $z \geq g_k(x)$

Subproblem

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

Obtain proof of optimality (solution of inference dual). Use same proof to deduce cost bounds for other assignments, yielding Benders cut.

Logic-Based Benders

- Fundamental concept: **inference duality**

Primal problem: optimization

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

Find **best** feasible solution by searching over **values of *x***.

Dual problem: Inference

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

Find a proof of optimal value *v** by searching over **proofs *P***.

In classical LP, the proof is a tuple of dual multipliers

Logic-Based Benders

- The proof that solves the dual in iteration k gives a bound $g_k(\bar{x})$ on the optimal value.
  - The same proof gives a bound $g_k(x)$ for other values of x.

Master problem

$$\min z$$
$$z \geq g_k(x) \quad \text{(Benders cuts)}$$
$$x \in D_x$$

Minimize cost z subject to bounds given by Benders cuts, obtained from values of x attempted in previous iterations k.

→ Trial value $\bar{x}$ that solves master

← Benders cut $z \geq g_k(x)$

Subproblem

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

Obtain proof of optimality (solution of inference dual). Use same proof to deduce cost bounds for other assignments, yielding Benders cut.

Logic-Based Benders

- Popular optimization duals are **special cases** of the inference dual.
  - Result from different choices of **inference method**.
  - For example....
    - Linear programming dual (gives **classical Benders cuts**)
    - Lagrangean dual
    - Surrogate dual
    - Subadditive dual

Logic-Based Benders Applications

- Planning and scheduling:
  - Machine allocation and scheduling
  - Steel production scheduling
  - Chemical batch processing (BASF, etc.)
  - Auto assembly line management (Peugeot-Citroën)
  - Allocation and scheduling of multicore processors (IBM, Toshiba, Sony)
  - Worker assignment in a queuing environment

Logic-Based Benders Applications

- Other scheduling
  - Lock scheduling
  - Shift scheduling
  - Permutation flow shop scheduling with time lags
  - Resource-constrained scheduling
  - Hospital scheduling
  - Optimal control of dynamical systems
  - Sports scheduling

Logic-Based Benders Applications

- Routing and scheduling
  - Vehicle routing
  - Home health care
  - Food distribution
  - Automated guided vehicles in flexible manufacturing
  - Traffic diversion around blocked routes
  - Concrete delivery

Logic-Based Benders Applications

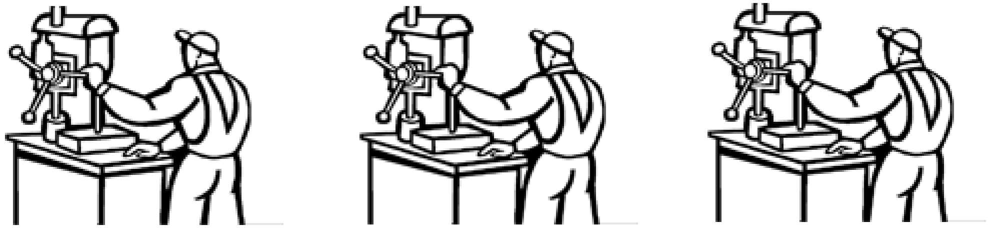
- Location and Design
  - Allocation of frequency spectrum (U.S. FCC)
  - Wireless local area network design
  - Facility location-allocation
  - Stochastic facility location and fleet management
  - Capacity and distance-constrained plant location
  - Queuing design and control

Logic-Based Benders Applications

- Other
  - Logical inference (SAT solvers essentially use Benders)
  - Logic circuit verification
  - Bicycle sharing
  - Service restoration in a network
  - Inventory management
  - Supply chain management
  - Space packing

Example: Machine Scheduling

- Assign tasks to machines.
- Then schedule tasks assigned to each machine.
  - Subject to time windows.
  - **Cumulative scheduling**: several tasks can run simultaneously, subject to resource limits.
  - Scheduling problem decouples into a separate problem for each machine.



Machine Scheduling

- Assign tasks in master, schedule in subproblem.
  - Combine mixed integer programming and constraint programming

Master problem

Assign tasks to resources to minimize cost.

Solve by mixed integer programming.

Trial assignment $\bar{x}$ →

← Benders cut $z \geq g_k(x)$

Subproblem

Schedule jobs on each machine, subject to time windows.

Constraint programming obtains proof of optimality (dual solution).

Use same proof to deduce cost for some other assignments, yielding Benders cut.

Machine Scheduling

- Objective function
  - Cost is based on task assignment only.

$$\text{cost} = \sum_{ij} c_{ij} x_{ij}, \quad x_{ij} = 1 \text{ if task } j \text{ assigned to resource } i$$

  - So cost appears only in the master problem.
  - Scheduling subproblem is a feasibility problem.

Machine Scheduling

- Objective function
  - Cost is based on task assignment only.

$$\text{cost} = \sum_{ij} c_{ij} x_{ij}, \quad x_{ij} = 1 \text{ if task } j \text{ assigned to resource } i$$

  - So cost appears only in the master problem.
  - Scheduling subproblem is a feasibility problem.

- Benders cuts
  - They have the form $\sum_{j \in J_i} (1 - x_{ij}) \geq 1, \text{ all } i$
  - where J_i is a set of tasks that create infeasibility when assigned to resource i.

Machine Scheduling

- Resulting Benders decomposition:

Master problem

$\min\ z$

$$z = \sum_{ij} c_{ij} x_{ij}$$

Benders cuts

Trial assignment $\bar{x}$

Benders cuts

$$\sum_{j \in J_i} (1 - x_{ij}) \geq 1,$$

for infeasible resources i

Subproblem

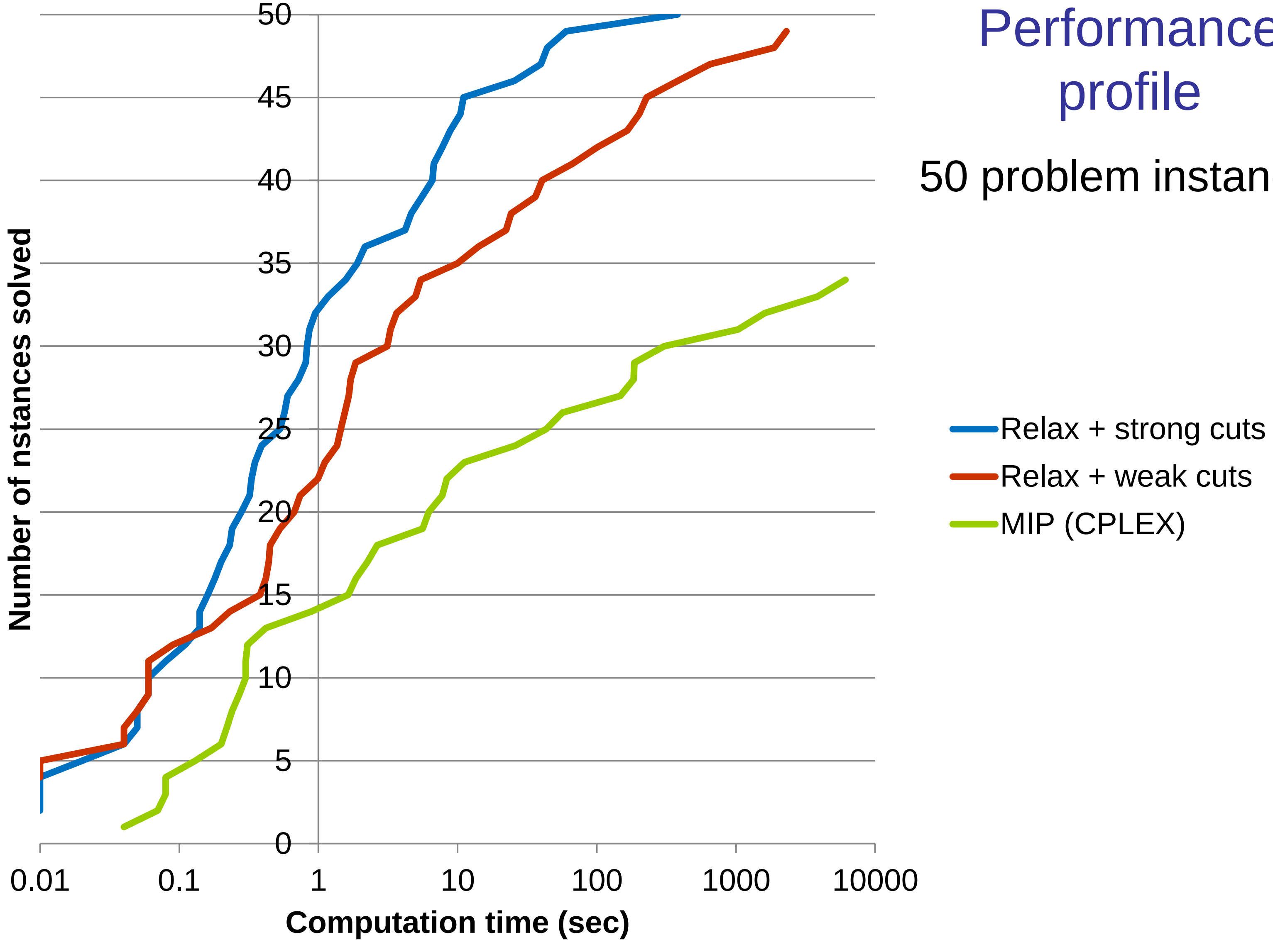
Schedule jobs on each resource.

Constraint programming may obtain proof of infeasibility on some resources (dual solution).

Use same proof to deduce infeasibility for some other assignments, yielding Benders cut.

Performance profile

50 problem instances



Extensions

- Other objective functions
  - Minimize makespan
  - Minimize number of late jobs
  - Minimize total tardiness
- Stronger Benders cuts
- Stronger relaxations

- Assume all release times are the same in cumulative scheduling subproblem...

Minimize Makespan

Master Problem: Assign tasks to resources

Formulate as MILP problem

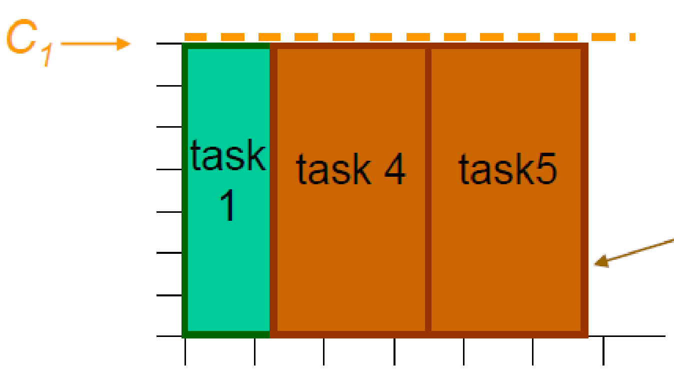
$$\min \quad M$$

makespan

$$\text{subject to} \quad \sum_i x_{ij} = 1, \quad \text{all } j$$

$$M \geq \frac{1}{C_i} \sum_j p_{ij} c_{ij} x_{ij}, \quad \text{all } i$$

Benders cuts



Relaxation of subproblem:
"Energy" of tasks provides lower bound on makespan.

Minimize Makespan

Benders cuts are based on:

Lemma. If we remove tasks 1, … *s* from a resource, the minimum makespan on that resource is reduced by at most

$$\sum_{j=1}^{s} p_{ij} + \max_{j \le s}\{d_j\} - \min_{j \le s}\{d_j\}$$

Assuming all deadlines d_j are the same, we get the Benders cut

$$M \ge M_{hi}^* - \sum_{j \in J_{hi}} (1 - x_{ij}) p_{ij}$$

Min makespan on resource *i* in last iteration

Minimize Number of Late Tasks

Master problem: Assign tasks to resources

= 1 if task *j* is assigned to resource *i*

$$\begin{aligned}
&\min && L \\
&\text{subject to} && \sum_i x_{ij} = 1, \quad \text{all } j \\
& && \text{Benders cuts} \\
& && \text{relaxation of subproblem} \\
& && x_{ij} \in \{0,1\}
\end{aligned}$$

Minimize Number of Late Tasks

Benders cuts

Lower bound on # late tasks on resource i

Min # late tasks on resource *i* (solution of subproblem)

$$L \geq \sum_i \hat{L}_{hi}$$

$$\hat{L}_{hi} \geq L^*_{hi} - L^*_{hi} \sum_{j \in J^0_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{L}_{hi} \geq L^*_{hi} - 1 - L^*_{hi} \sum_{j \in J^1_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{L}_{hi} \geq 0, \quad \text{all } i$$

subset of J_{hi} for which min # late tasks is still L_{hi}^* (found by heuristic that repeatedly solves subproblem on resource *i*)

Minimize Number of Late Tasks

Benders cuts

$$L \geq \sum_i \hat{L}_{hi}$$

$$\hat{L}_{hi} \geq L^*_{hi} - L^*_{hi} \sum_{j \in J^0_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{L}_{hi} \geq L^*_{hi} - 1 - L^*_{hi} \sum_{j \in J^1_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{L}_{hi} \geq 0, \quad \text{all } i$$

Min # late tasks on resource *i* (solution of subproblem)

subset of J_{hi} for which min # late tasks is still L_{hi}^* (found by heuristic that repeatedly solves subproblem on resource *i*)

Smaller subset of J_{hi} for which min # late tasks is $L_{hi}^* - 1$ (found while running same heuristic)

Minimize Number of Late Tasks

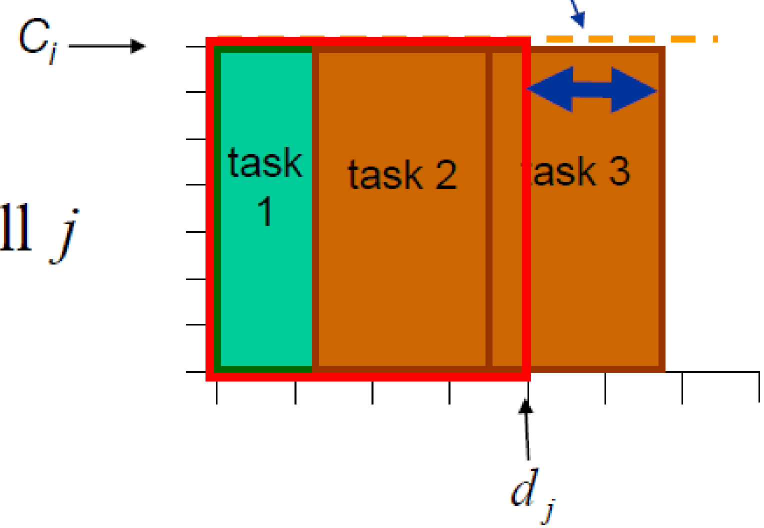
Relaxation of subproblem

Lower bound on (makespan – latest deadline)

$$L \ge \sum_i L_i$$

$$L_i \ge \frac{\dfrac{1}{C_i} \displaystyle\sum_{k \in J(d_j)} c_{ik} p_{ik} x_{ik} - d_j}{\displaystyle\max_{k \in J(d_j)} \{p_{ik}\}}, \quad \text{all } j$$

Max processing time



Minimize Total Tardiness

Master problem: assign tasks to resources

$$\begin{aligned} \min \quad & L \\ \text{subject to} \quad & \sum_i x_{ij} = 1, \quad \text{all } j \\ & \text{Benders cuts} \\ & \text{relaxation I of subproblem} \\ & \text{relaxation II of subproblem} \\ & x_{ij} \in \{0,1\} \end{aligned}$$

x_{ij} = 1 if task j is assigned to resource i

Minimize Total Tardiness

Benders cuts

Lower bound on tardiness for resource *i*

Min tardiness on resource *i* (solution of subproblem)

$$T \ge \sum_i \hat{T}_{hi}$$

$$\hat{T}_{hi} \ge T^*_{hi} - T^*_{hi} \sum_{j \in J_{hi}} (1 - x_{ij}), \quad \text{all } i$$

To reduce tardiness on resource *i*, must remove one of the tasks assigned to it.

$$\hat{T}_{hi} \ge T^0_{hi} - T^0_{hi} \sum_{j \in J_{hi} \setminus Z_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{T}_{hi} \ge 0, \quad \text{all } i$$

Minimize Total Tardiness

Benders cuts

$$T \geq \sum_i \hat{T}_{hi}$$

$$\hat{T}_{hi} \geq T^*_{hi} - T^*_{hi} \sum_{j \in J_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{T}_{hi} \geq T^0_{hi} - T^0_{hi} \sum_{j \in J_{hi} \setminus Z_{hi}} (1 - x_{ij}), \quad \text{all } i$$

$$\hat{T}_{hi} \geq 0, \quad \text{all } i$$

To reduce tardiness below T^0_{hi} on resource *i*, must remove one of the tasks in $J_{hi} \setminus Z_{hi}$

Set of tasks that can be removed, one at a time from resource *i* without reducing min tardiness.

Min tardiness on resource *i* when all tasks in Z_{hi} are removed *simultaneously*.

Minimize Total Tardiness

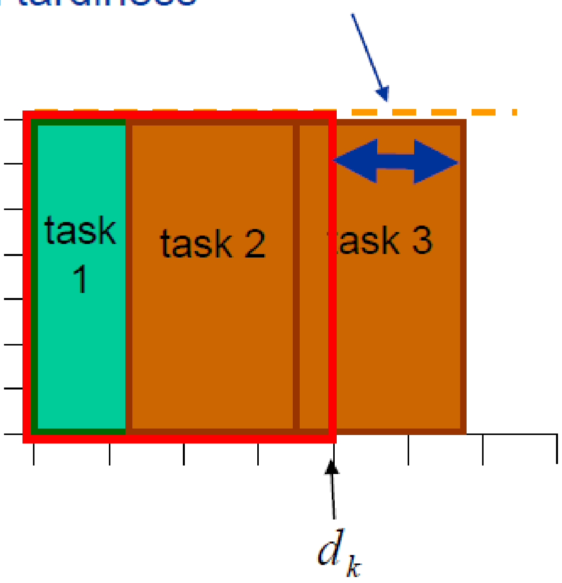
Subproblem relaxation I

Lower bound on total tardiness for resource *i*

Lower bound on total tardiness

$$T \geq \sum_i T_i$$

$$T_i \geq \frac{1}{C_i} \sum_{j \in J(d_k)} c_{ij} p_{ij} x_{ij} - d_k, \text{ all } k$$



Minimize Total Tardiness

Subproblem relaxation II

Lemma. Consider a min tardiness problem that schedules tasks 1, …, *n* on resource *i*, where $d_1 \le \ldots \le d_n$. The min tardiness T^* is bounded below by

$$\bar{T} = \sum_{k=1}^{n} \bar{T}_k$$

where

$$\bar{T}_k = \left(\frac{1}{C_i} \sum_{j=1}^{k} p_{i\pi_i(j)} c_{i\pi_i(j)} - d_k \right)^+$$

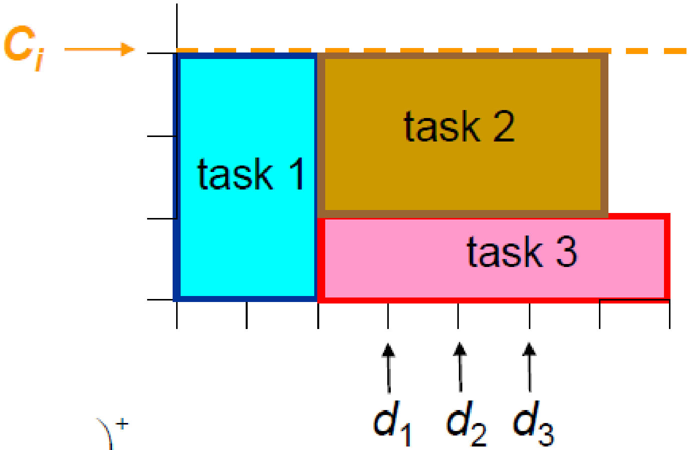
and π is a permutation of 1, …, *n* such that

$$p_{\pi_i(1)} c_{\pi_i(1)} \le \cdots \le p_{\pi_i(n)} c_{\pi_i(n)}$$

Minimize Total Tardiness

Example of Lemma

j	d_j	p_{ij}	c_{ij}	$p_{ij}c_{ij}$
1	3	2	3	6
2	4	4	2	8
3	5	5	1	5



$$\overline{T}_1 = \left(\frac{1}{C_i}(p_{i3}c_{i3}) - d_1\right)^+ = \left(\frac{1}{3}(5) - 3\right)^+ = 0$$

$$\overline{T}_2 = \left(\frac{1}{C_i}(p_{i3}c_{i3} + p_{i1}c_{i1}) - d_2\right)^+ = \left(\frac{1}{3}(5+6) - 4\right)^+ = 0$$

$$\overline{T}_3 = \left(\frac{1}{C_i}(p_{i3}c_{i3} + p_{i1}c_{i1} + p_{i2}c_{i2}) - d_3\right)^+ = \left(\frac{1}{3}(5+6+8) - 5\right)^+ = 4/3$$

$$\text{Lower bound on tardiness} = \left\lceil \overline{T}_1 + \overline{T}_2 + \overline{T}_3 \right\rceil = \lceil 4/3 \rceil = 2$$

Min tardiness = 4

Minimize Total Tardiness

Writing relaxation II

From the lemma, we can write the relaxation

$$T \geq \sum_i \sum_{k=1}^{n} T'_{ik} x_{ik}$$

where $T'_{ik} \geq \frac{1}{C_i} \sum_{j=1}^{k} p_{i\pi_i(j)} c_{i\pi_i(j)} x_{i\pi_i(j)} - d_k$

To linearize this, we write $T \geq \sum_i \sum_{k=1}^{n} T_{ik}$

and $T_{ik} \geq \frac{1}{C_i} \sum_{j=1}^{k} p_{i\pi_i(j)} c_{i\pi_i(j)} x_{i\pi_i(j)} - d_k - (1 - x_{ik}) M_{ik}$

where $M_{ik} = \frac{1}{C_i} \sum_{j=1}^{k} p_{i\pi_i(j)} c_{i\pi_i(j)} - d_k$

Application: Home Health Care

- General home health care problem.
  - Assign **aides** to homebound **patients**.
    - …subject to constraints on aide qualifications and patent preferences.
    - One patient may require a team of aides.
  - **Route** each aide through assigned patients, observing **time windows**.
    - …subject to constraints on hours, breaks, etc.

Home Health Care

- A large industry, and **rapidly growing**.
  - Roughly as large as all courier and delivery services.

Projected Growth of Home Health Care Industry

	2014	2018
U.S. revenues, $ billions	75	150
World revenues, $ billions	196	306

Increase in U.S. Employment, 2010-2020

Home health care industry	70%
Entire economy	14%

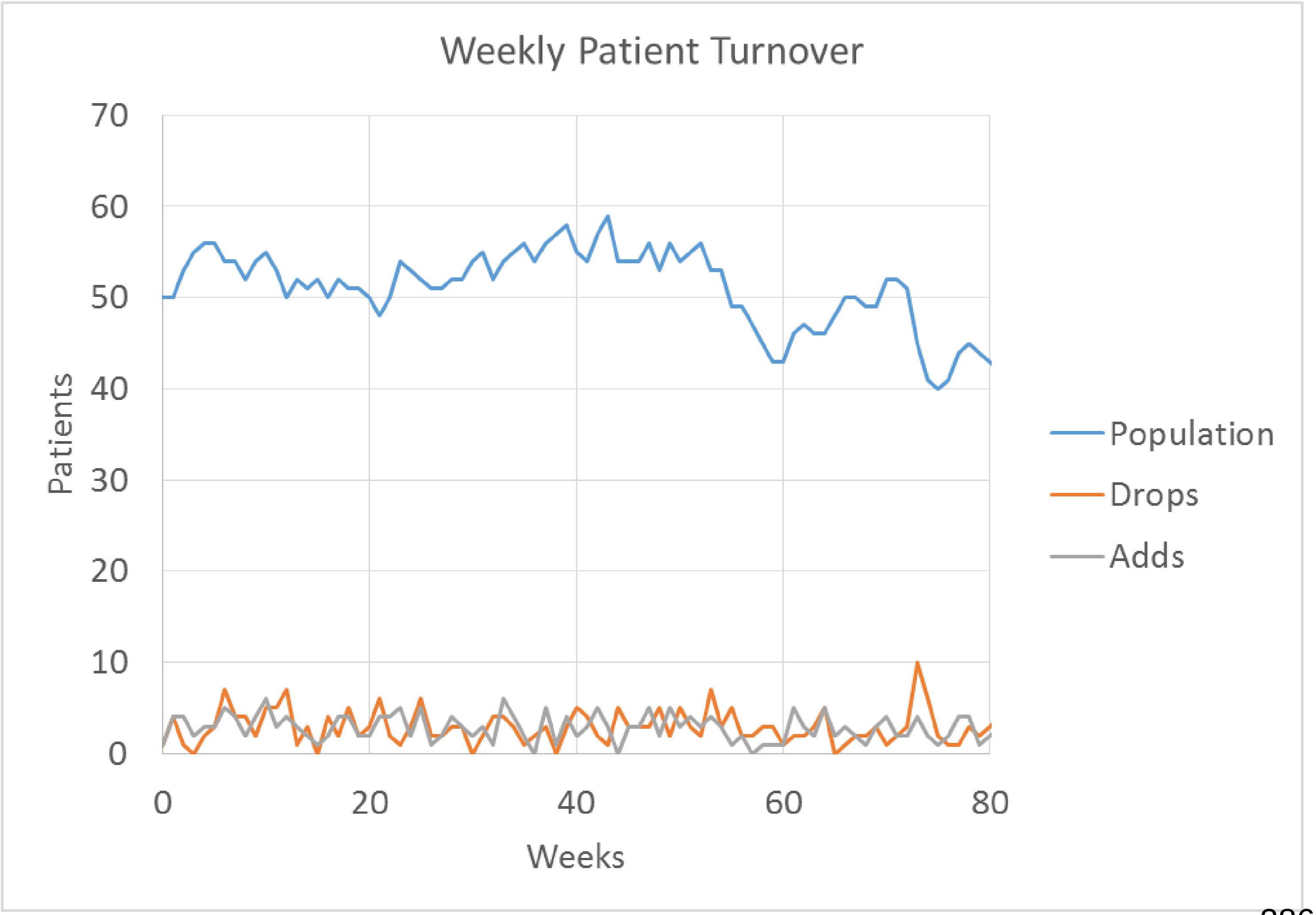
Home Health Care

- Advantages of home healthcare
  - Lower cost
    - Hospital & nursing home care is very expensive.
  - No hospital-acquired infections
    - Less exposure to superbugs.
  - Preferred by patients
    - Comfortable, familiar surroundings of home.
    - Sense of control over one's life.
  - Supported by new equipment & technology
    - IT integration with hospital systems.
    - Online consulting with specialists.

Home Hospice Care

- Distinguishing characteristics
  - Personal & household services
  - Regular weekly schedule
    - For example, Mon-Wed-Fri at 9 am.
  - Same aide each visit
  - Long planning horizon
    - Several weeks
  - Rolling schedule
    - Update schedule as patient population evolves.

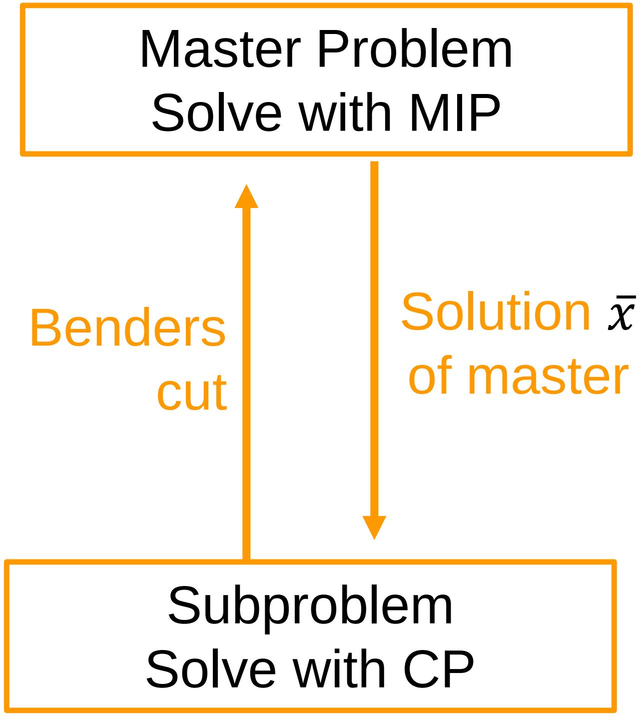
Home Hospice Care



5-8% weekly turnover

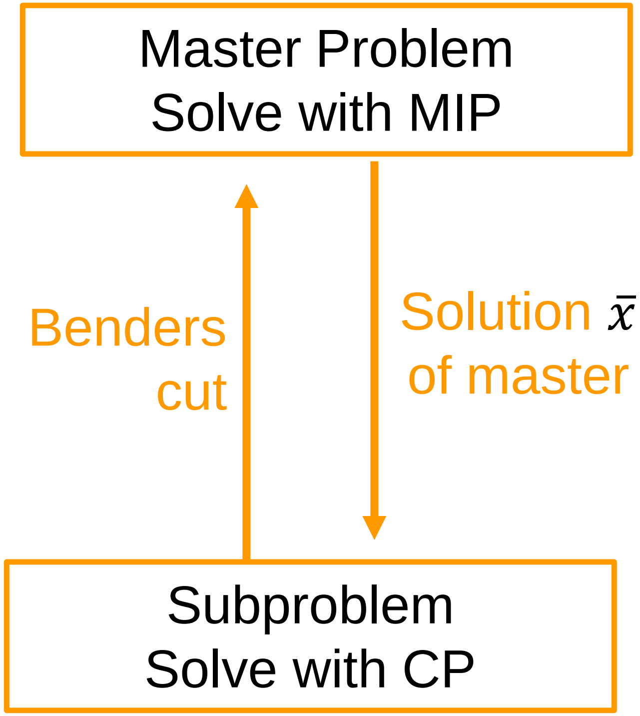
Home Hospice Care

- Solve with Benders decomposition.
  - **Assign aides to patients** in master problem.
    - Maximize number of patients served by a given set of aides.



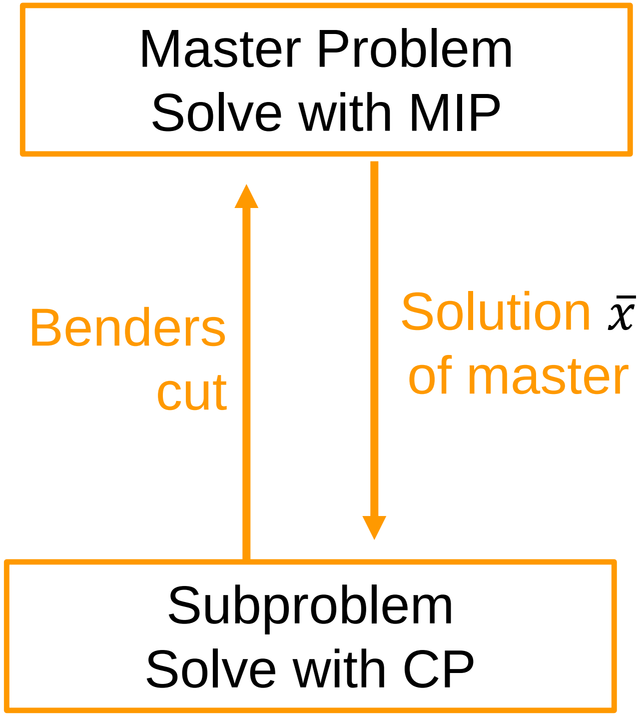
Home Hospice Care

- Solve with Benders decomposition.
  - **Assign aides to patients** in master problem.
    - Maximize number of patients served by a given set of aides.
  - **Schedule home visits** in subproblem.
    - Cyclic weekly schedule.
    - Visit each patient same time each day.
    - No visits on weekends.



Home Hospice Care

- Solve with Benders decomposition.
  - **Assign aides to patients** in master problem.
    - Maximize number of patients served by a given set of aides.
  - **Schedule home visits** in subproblem.
    - Cyclic weekly schedule
    - Visit each patient same time each day.
    - No visits on weekends.
  - Subproblem **decouples** into a scheduling problem for each aide



Master Problem

= 1 if patient *j* scheduled

= 1 if patient *j* assigned to aide *i*

= 1 if patient *j* assigned to aide *i* on day *k*

Required number of visits per week

$$\max \sum_j \delta_j$$

$$\sum_i x_{ij} = \delta_j, \quad \text{all } j$$

$$\sum_{i,k} y_{ijk} = v_j \delta_j, \quad \text{all } j$$

$$y_{ijk} \leq x_{ij}, \quad \text{all } i, j, k$$

Spacing constraints on visit days

Benders cuts

Relaxation of subproblem

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

Master Problem

- For a rolling schedule:
  - Schedule **new patients**, drop **departing patients** from schedule.
    - Provide continuity for remaining patients as follows:
  - Old patients served by **same aide** on **same days**.
    - Fix $y_{ijk} = 1$ for the relevant aides, patients, and days.

Subproblem

Simplified routing & scheduling problem for aide *i*

*n*th patient in sequence

Patients assigned to aide *i*

$$\text{all-different}\{\pi_{k\nu} \mid \nu = 1, \ldots, |P_i|\}$$

$$[s_j, s_j + p_j] \subseteq [r_j, d_j]$$

$$s_{\pi_{k\nu}} + p_{\pi_{k\nu}} + t_{\pi_{k\nu}\pi_{k,\nu+1}} \leq s_{\pi_{k,\nu+1}}, \quad \text{all } k, \nu$$

start time

Visit duration

Travel time

Modeled with interval variables in CP solver

Benders Cuts

- Generate a cut for each infeasible scheduling problem.
  - Solution of subproblem inference dual is a **proof** of infeasibility.
    - The proof may show **other** patient assignments to be infeasible.
    - Generate **nogood cut** that rules out these assignments.

Benders Cuts

- Generate a cut for each infeasible scheduling problem.
  - Solution of subproblem inference dual is a **proof** of infeasibility.
    - The proof may show **other** patient assignments to be infeasible.
    - Generate **nogood cut** that rules out these assignments.
  - Unfortunately, we **don't have access** to infeasibility proof in CP solver.

Benders Cuts

- So, strengthen the nogood cuts heuristically.
  - Find a smaller set of patients that create infeasibility…
    - …by re-solving the each infeasible scheduling problem repeatedly.

$$\sum_{j \in \boxed{\bar{P}_i}} (1 - y_{ijk}) \geq 1$$

Reduced set of patients whose assignment to aide *i* creates infeasibility

Benders Cuts

- Include relaxation of subproblem in the master problem.
  - Necessary for good performance.
  - Use **time window relaxation** for each scheduling problem.
  - Simplest relaxation for aide *i* and day *k*:

$$\sum_{j \in \boxed{J(a,b)}} p_j y_{ijk} \le b - a$$

Set of patients whose time window fits in interval [*a*, *b*].

Can use several intervals.

Subproblem Relaxation

- This relaxation is very weak.
  - Doesn't take into account travel times.

Subproblem Relaxation

- This relaxation is very weak.
  - Doesn't take into account travel times.
- Improved relaxation.
  - **Basic idea:** Augment visit duration p_j with travel time to (or from) location j from **closest** patient or aide home base.

Subproblem Relaxation

- This relaxation is very weak.
  - Doesn’t take into account travel times.
- Improved relaxation.
  - **Basic idea:** Augment visit duration p_j with travel time to (or from) location j from **closest** patient or aide home base.
  - This is **weak** unless most assignments are **fixed**.
    - As in rolling schedule.

Subproblem Relaxation

- This relaxation is very weak.
  - Doesn't take into account travel times.
- Improved relaxation.
  - **Basic idea:** Augment visit duration p_j with travel time to (or from) location j from **closest** patient or aide home base.
  - This is **weak** unless most assignments are **fixed**.
    - As in rolling schedule.
  - Find intervals that yield tightest relaxation
    - Short intervals that contain many time windows.

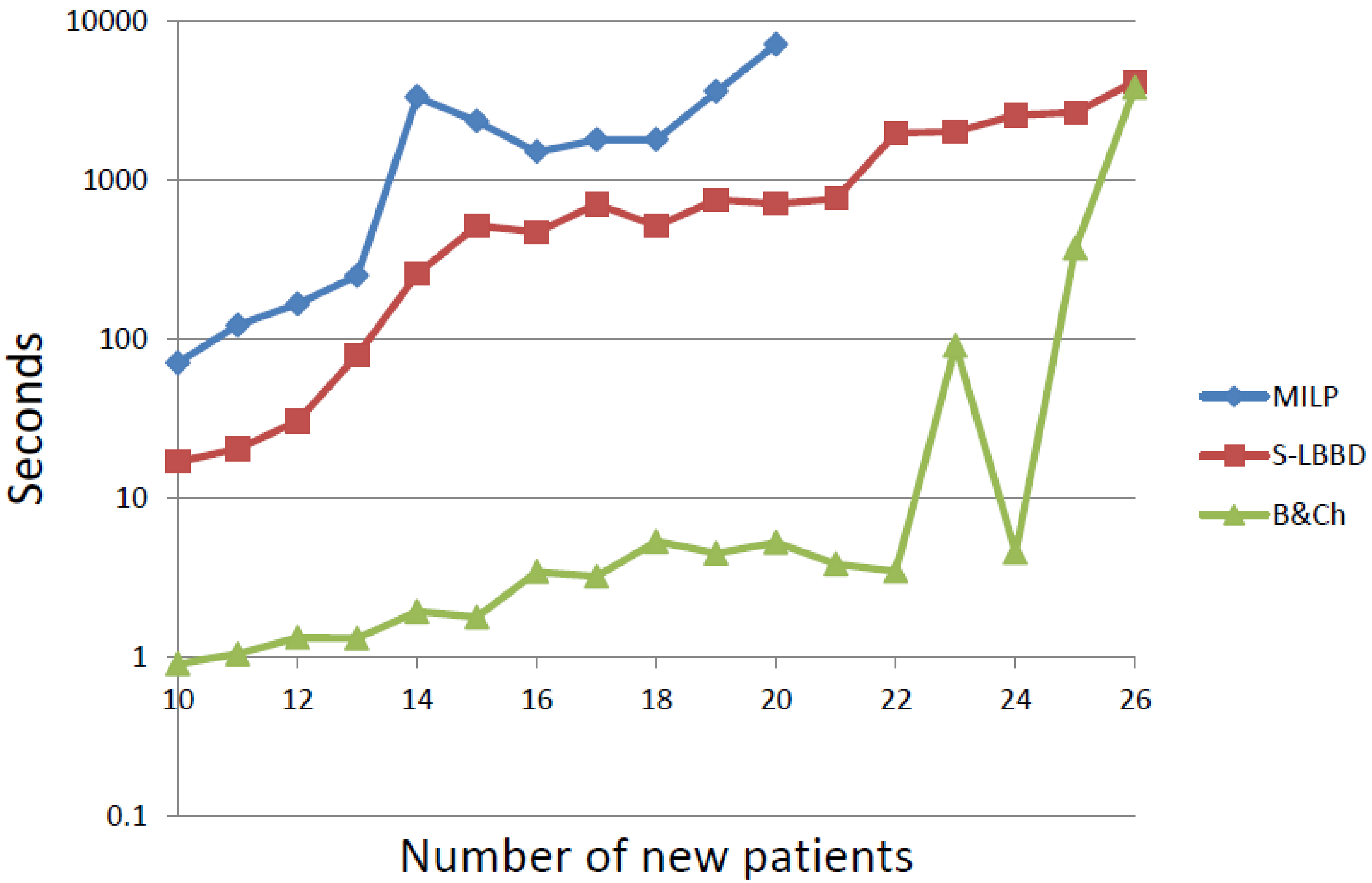
Branch and Check

- A variation of logic-based Benders
  - Solve master problem only **once**, by branching.
  - At feasible nodes, solve subproblem to obtain Benders cut.
  - **Not the same** as branch & cut.
- Use when master problem is the bottleneck
  - Subproblem solves much faster than master problem.

Computational Tests

- Original real-world dataset
  - 60 home hospice patients
    - 1-5 visits per week (not on weekends)
  - 18 health care aides with time windows
  - Actual travel distances
- Solver
  - **LBBD:** Hand-written code manages MIP & CP solvers
    - SCIP + Gecode
  - **Branch & check:** Use constraint handler in SCIP
    - SCIP + Gecode
  - **MIP:** SCIP
    - Modified multicommodity flow model of VRPTW

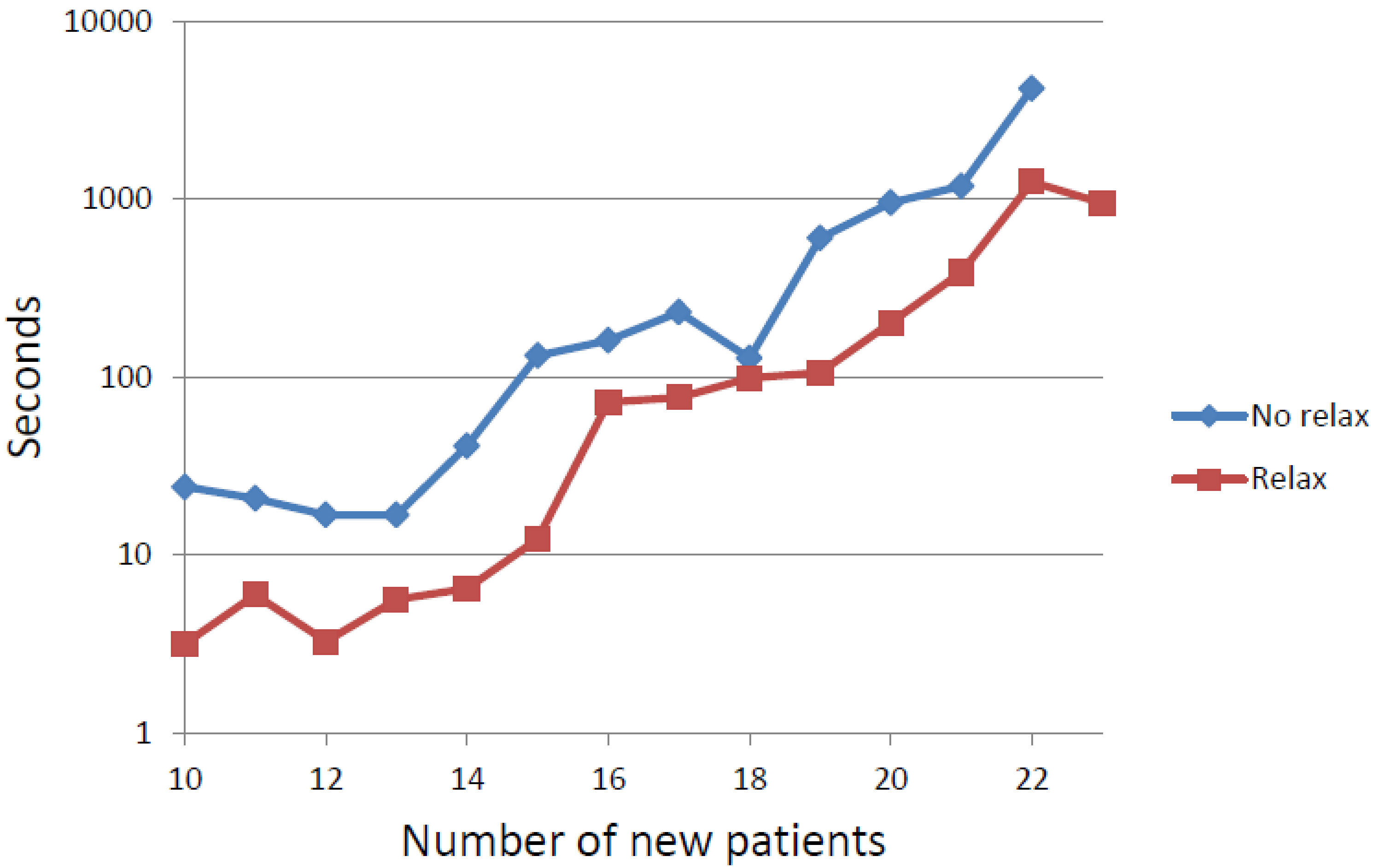
Computation time, fewer visits per week



Computational Tests

- Practical implications
  - Branch & check **scales up** to realistic size
    - One month advance planning for original 60-patient dataset
    - Assuming 5-8% weekly turnover
    - Much faster performance for modified dataset
  - Advantage of **exact** solution method
    - We know **for sure** whether existing staff will cover projected demand.

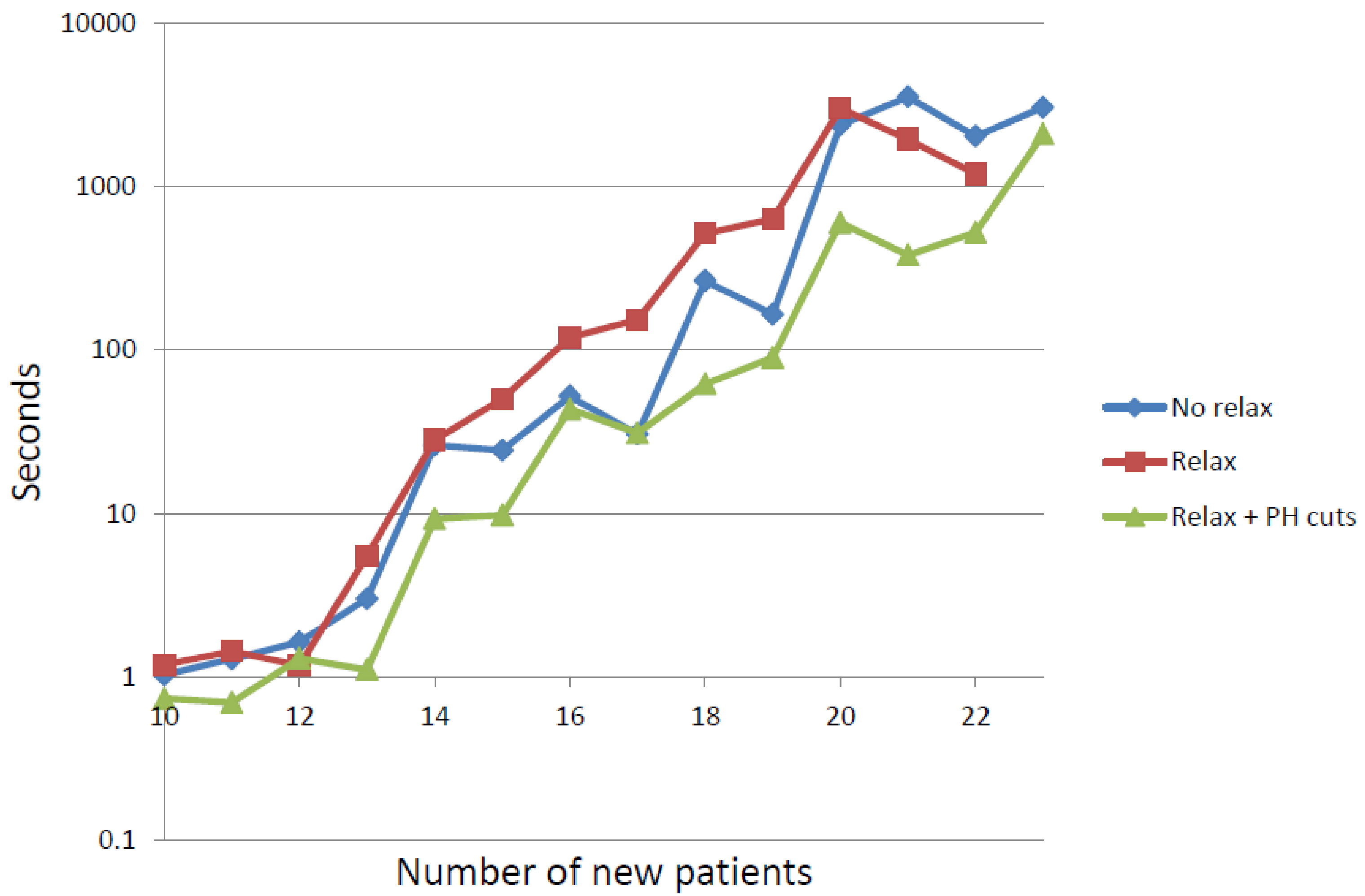
Effect of time window relaxation
Standard LBBD
Original problem data



Effect of time window relaxation and primal heuristic cuts

Branch & check

Original problem data



Computational Tests

- Rasmussen instances
  - From 2 Danish municipalities
    - One-day problem
    - We extended it to 5 days with same schedule each day
    - Reduce number of patients to 30, so MIP has a chance
  - Solve problem from scratch
    - No rolling schedule
  - Two objective functions
    - **Weighted:** Minimize weighted average of travel cost, matching cost (undesirability of assignment), uncovered patients.
    - **Covering:** Minimize number of uncovered patients (same as ours)

Table 6 Solution time (s) for modified Rasmussen instances

Instance	Patients	Crews	Weighted objective MILP	Weighted objective LBBD	Weighted objective B&Ch	Covering objective MILP	Covering objective LBBD	Covering objective B&Ch
hh	30	15	*	3.16	**1.41**	*	**23.3**	441
ll1	30	8	*	1.74	**0.43**	*	108	**1.41**
ll2	30	7	2868	1.56	**0.32**	*	**1.38**	6.45
ll3	30	6	1398	2.16	**0.30**	*	**3.07**	5.98

*Computation time exceeded one hour.

Table 6 Solution time (s) for modified Rasmussen instances

Instance	Patients	Crews	Weighted objective MILP	LBBD	B&Ch	Covering objective MILP	LBBD	B&Ch
hh	30	15	*	3.16	**1.41**	*	**23.3**	441
ll1	30	8	*	1.74	**0.43**	*	108	**1.41**
ll2	30	7	2868	1.56	**0.32**	*	**1.38**	6.45
ll3	30	6	1398	2.16	**0.30**	*	**3.07**	5.98

*Computation time exceeded one hour.

Standard LBBD tends to be better when subproblem consumes most of the solution time in branch & check

Table 2 Percent of solution time devoted to subproblem

Instances	S-LBBD Avg	S-LBBD Max	B&Ch Avg	B&Ch Max
Original 60-patient instances	0.1	0.2	1.4	3.9
Narrow time windows	0.1	0.1	2.8	6.0
Fewer visits per patient	0.0	0.1	1.7	3.5
Rasmussen, weighted objective	0.4	0.8	6.3	13.6
Rasmussen, covering objective	1.2	1.5	85.6	99.7

Computational Tests

- LBBD can scale up despite sequence-dependent costs…
  - …especially when computing a **rolling** schedule
    - Time window relaxation is tight enough in this case
  - Routing & scheduling problems remain small as patient population increases
    - The 4-index MIP variables explode as the population grows
    - …even for a rolling schedule

Computational Tests

- LBBD can scale up despite sequence-dependent costs…
  - …especially when computing a **rolling** schedule
    - Time window relaxation is tight enough in this case
  - Routing & scheduling problems remain small as patient population increases
    - The 4-index MIP variables explode as the population grows
    - …even for a rolling schedule
- However…
  - LBBD not designed for temporal dependencies
    - As when multiple aides must visit a patient simultaneously.
    - Unclear how much performance degrades in this case.

Software

For integration of CP and MIP

- ECLiPSe
  - Exchanges information between ECLiPSe solver, Xpress-MP
- OPL Studio
  - Combines CPLEX MIP and CP Optimizer with script language
- Mosel
  - Combines Xpress-MP, Xpress-Kalis with low-level modeling
- BARON
  - Global optimization with relaxation + domain reduction
- SIMPL
  - Full integration with high-level modeling (prototype)
- SCIP
  - Combines MIP and CP-based propagation
- MiniZinc
  - High-level modeling with solver integration, including logic-based Benders


THE END